香港城市大學
City University of Hong Kong

Department of Computer Science

BSCCS Final Year Project Report
2024-2025

24CS116

Deep Learning-Based Posture monitoring App

(Volume 1 of 1)

Student Name : **TWAN Tsz Yin**

Student No. : **57149200**

Programme Code : **BSCEGU4**

For Official Use Only

Supervisor : **Prof XU, Weitao**

1st Reader : **Prof LAM, Kam Yiu**

2nd Reader : **Prof WANG, Lusheng**

Student Final Year Project Declaration

I have read the project guidelines and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I hereby declare that the work I submitted for my final year project, entitled:

Deep Learning-Based Posture monitoring App

does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Project Guidelines.

Student Name: TWAN Tsz Yin Signature: 

Student ID: 57149200 Date: March 30, 2025

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Prof. Xu Weitao, for his feedback and support throughout this project. His encouragement, the freedom he provided me to explore and design the entire application, as well as his affirmation of my literature review and experimental results, have been essential in shaping the success of this work.

I would also like to extend my deepest appreciation to the City University of Hong Kong and the Department of Computer Science for providing me with the resources and environment necessary to carry out this project. The academic support and access to facilities have contributed greatly to the completion of this work.

Abstract

With the rising popularity of weight training among people who seek to enhance their physical appearance and address body dissatisfaction, there is also a significant increase in injury rates. These injuries are mainly caused by a lack of knowledge about proper weight training techniques and an underestimation of the importance of correct posture. However, to obtain such knowledge and awareness can be costly (hiring personal trainers) or delayed (searching online).

Therefore, the purpose of this project is to develop a real-time posture monitoring app for weight training that can contribute to the training effectiveness of individuals and reduce the risk of injury. The app uses the front camera of a mobile device to record body posture during exercises, and provide real-time feedback on posture accuracy.

Pose estimation, an application of convolutional neural networks, enables the app to monitor posture by pinpointing human joints in real-time. By utilizing existing lightweight 2D pose estimation models optimized for mobile, the app can accurately extract joint coordinates from frames captured by the phone cameras. These coordinates will then feed into a posture classifier for analysis and determine the correctness of posture.

This project consists of two stages of evaluation: the performance evaluation of pose estimation models and the performance evaluation of classification algorithms.

In the first stage, computational intensiveness and human joint prediction accuracy of some existing pose estimation models (MoveNet, BlazePose and YOLO11n-pose) were compared, and the best performed model (BlazePose Full) was selected for implementation in the posture monitoring application.

In the second stage, the accuracies of classifying posture correctness using trigonometrically computed joint angles and using various machine learning algorithms (e.g.

KNN, Random Forest, and Logistic Regression), and the approach of using trigonometrically computed joint angles was selected for posture classification tasks due to its satisfactory accuracy and scalability.

Table of Contents

1. Introduction

1.1 Trend of Weight Training

Body dissatisfaction has played a significant role in people's quality of life, where Hysi [1] found that people's self-esteem is negatively affected when their actual physical form differs from the ideal body image. As a result, participating in gym activities has become a popular option for those seeking to transform their bodies into an ideal form, thus boosting their self-esteem. Thompson et al. [2] conducted a survey on fitness trends in 2023 and discovered that out of the top three most popular fitness activities, activities involving weights occupied two of them (see Figure 1).

Rank	Trend
1	Wearable technology
2	Strength training with free weights
3	Body weight training
4	Fitness programs for older adults
5	Functional fitness training
6	Outdoor activities
7	High-intensity interval training (HIIT)
8	Exercise for weight loss
9	Employing certified fitness professionals
10	Personal training

Figure 1. Top 10 worldwide fitness trends for 2023 [2].

1.2 Importance of Correct Posture in Weight Training

However, gym beginners and enthusiasts often overlook the importance of correct posture during workouts, leading to ineffective repetitions or even resulting in injuries, especially during weight training. Cuthbertson-Moon et al. [3] studied the gym injury claims reported over the past 10 years in New Zealand, where they discovered that gym injuries caused by lifting, carrying, and straining accounted for up to 49% of the 331,343 total injury claims. They concluded that the contributing factors to these injuries include a lack of accessibility to

exercise guidelines and improper posture. Bukhary et al. [4] also believe that poor posture and overtraining are the two contributing factors to musculoskeletal injuries for weightlifters. These conclusions reflect the importance of possessing weight training knowledge, such as maintaining correct posture, which is essential for preventing injuries.

1.3 Motivation

With the rise in popularity of weight training exercises, the number of weight training injury cases is likely to increase due to their high injury rate. The most common approaches for receiving guidance on proper weight training posture are using online platforms such as YouTube or hiring a personal trainer. Online platforms provide demonstrations from gym professionals, but they lack real-time and personalized feedback for individuals, which can be misleading and may worsen the situation. Although personal trainers provide real-time, personalized feedback, they are affordable only to a minority of financially capable individuals.

Although there are existing applications that utilize lightweight PE models for posture monitoring and exercise tracking, they either fail to provide real-time posture correctness feedback or lack portability.

For example, a mobile application available on Apple Store called BeOneSports can only perform posture monitoring after the workout session, which is done by sending the video recordings of users' workout to the PE model for keypoint extraction [5]. As a result, BeOneSports is unable to provide the real-time feedback during workout session that is essential for preventing injuries and ensuring workout efficiency.

In contrast, the GitHub project "Fitness Trainer Pose Estimation" has demonstrated the ability to track user's posture and provide real-time feedback on posture correctness, but the project was designed to execute on computers rather than mobile devices [6]. Therefore, users

may face issues such as compatibility and the use of computationally intense PE model when applied to mobile operating systems.

Therefore, there is a need for a solution that is easy-to-use, affordable, and able to provide personalized and situational real-time feedback to help individuals maintain their posture during workouts and prevent injuries.

1.4 Background of Pose Estimation

Convolutional Neural Network models (CNN), as a type of neural network that is widely implemented for image processing tasks such as facial recognition and autonomous driving, can also play its role in the fitness domain to improve the safety and effectiveness of individuals. Novatchkov et al. [7] have also highlighted the potential use of AI in automating the analysis and feedback in the sports and weight training domain.

Pose Estimation (PE) is one of the applications of CNN that is specifically trained to track the movement and posture of the human body in real-time with great accuracy. The task of PE is to locate the human body and pinpoint the body parts that have significant contribution to the human posture given an image, such as knees, elbows, and wrists, as coordinates [8]. With these coordinates highlighted on the human body, changes in angles at human joints and the movements of body parts can be measured, thus achieving posture monitoring purposes.

1.5 Project Objectives and Overview

The purpose of this project is to explore the possibility, efficiency, and feasibility of applying CNN in real-life applications that rely purely on the computational power of mobile devices. For the final deliverable, one of the existing lightweighted PE models will be apply to a mobile application to provide a portable solution for gym beginners and gym enthusiasts,

assist them during some of the weight training workouts such as straight arm pulldowns and skull crusher. By constantly feeding image inputs to the model using the front camera of the mobile device with keypoints analytics during training, real-time posture feedback can be provided to users.

2. Literature Review

The literature review section will start with discussing some current mobile-friendly PE models and state-of-the-art PE models, including their expected outputs and general model architecture. Then, some existing applications that have utilized PE in the fitness domain will be explored, with their approaches potentially being applied in this project.

2.1 2D Lightweight Pose Detection with MoveNet

MoveNet [9] is an open-source CNN model developed by Google for PE, it produces predicts 17 human keypoints of the full body (see Figure 2), and has demonstrated satisfactory accuracy in real time for completing PE tasks with low computational requirements.

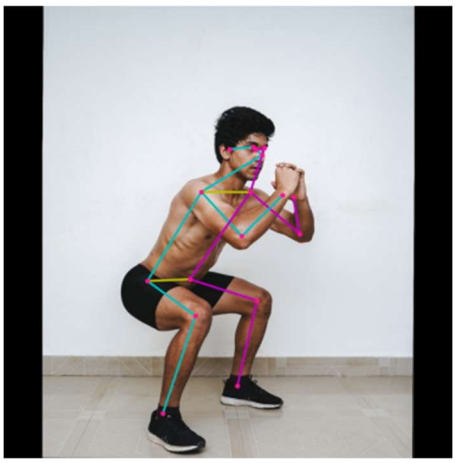
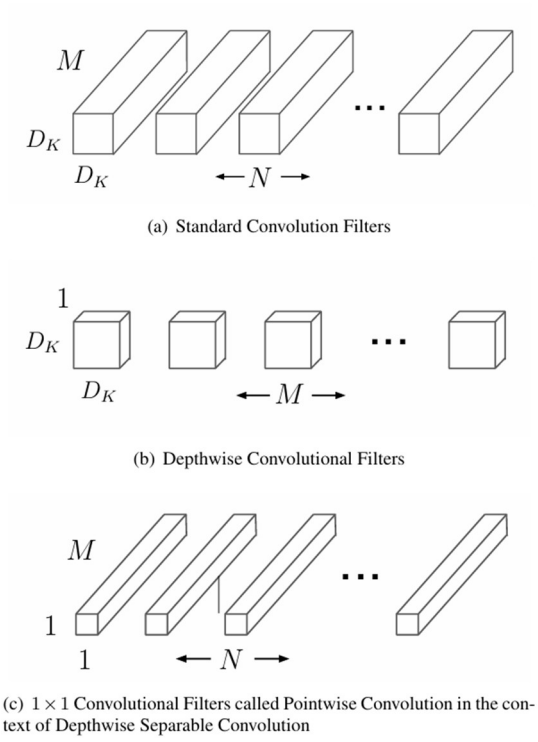


Figure 2. Keypoint landmarks outputted from MoveNetV2 [9].

MoveNet used MobileNetV2 developed by Sandler et al. [10] as its image feature extractor, featuring depthwise separable convolutions, inverted residuals and linear bottlenecks. Depthwise separable convolutions [11] separate the convolution process into lightweighted depthwise convolutions that apply a single 3x3 filter separately to each input channel for spatial filtering, and pointwise convolutions that use 1x1 convolutions to combine and transform the channel dimensions of the output (see Figure 3), thereby achieving a significant reduction in the number of parameters and computational operations needed comparing to the standard convolutional approach. For linear bottlenecks, the model performs pointwise convolutions to control channel dimensions, with a linear activation in the bottleneck layer, computational demand can be reduced while preserving important information in the network layers (see Figure 4). With inverted residuals, bottleneck blocks undergo input expression first to increase the channel capacity for feature extraction, then compression later on to preserve the most essential features, also with direct shortcuts connecting between the bottlenecks to avoid vanishing gradients (see Figure 5).



(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 3. Standard convolution layer (a) is replaced by depthwise convolution in (b) and

pointwise convolution in (c) [11].

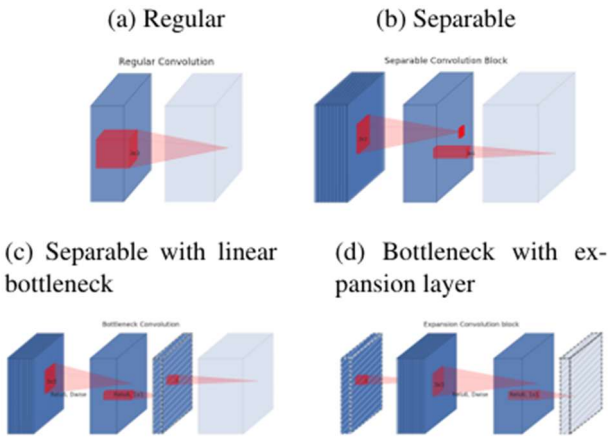


Figure 4. Evaluation of separable convolution blocks [9].

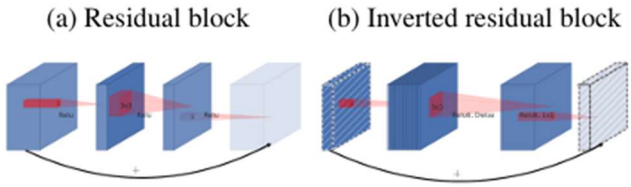


Figure 5. Comparison of residual block and inverted residual [11].

The extracted multi-layer features from MobileNetV2 are then passed to a feature pyramid network (FPN) [12] decoder for multi-scale feature fusion and refinement. The FPN first uses a bottom-up pathway that performs a standard feedforward process of MobileNetV2, where the feature details are captured and form a feature pyramid with feature maps at different scales, with the deeper layers expected to contain stronger semantic information. The extracted multi-layer features also undergo a top-down pathway that upsamples the spatially coarser, high-level feature maps from the deeper layers of the pyramid to higher resolutions, aiming to restore finer spatial details. The bottom-up pathway and top-down pathway are linked with lateral connections, this merges upsampled top-down features with higher-resolution bottom-up features, which undergo a 1×1 convolution for compatibility (see Figure 6). This merging

process allows refinement of multi-scale feature maps by combining semantic richness with spatial details, enhancing the accuracy and robustness of keypoint detection and localization.

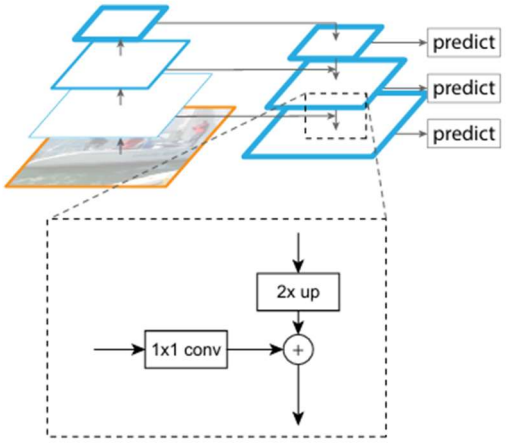


Figure 6. The lateral connection and the top-down pathway, merged by addition [12].

After the FPN finishes generating the refined multi-scale feature maps, they are passed to the prediction head of MoveNet for keypoint prediction and localization, which is based on CenterNet [13]. CenterNet is initially designed to use a heatmap prediction approach to predict the center points of objects, but in MoveNet, CenterNet uses offsets from the center point of the detected human body as a representation of the joint keypoints. To predict each joint keypoint, the center point of the human body is directly regressed to the corresponding offset, which indicates how far the joint keypoint is from the center point. Heatmaps are also generated for each keypoint, providing an approximation of the keypoint locations for prediction (see Figure 7).

keypoint heatmap [C] local offset [2] object size [2]

Figure 7. Object detection (top) and joint keypoints detection (bottom). The number in the brackets indicates the output channels [13].

MoveNet currently provides two variations—MoveNet Lightning and MoveNet Thunder [9]. Although the detailed parameter numbers for these two variations are not specified, they are claimed to be capable of performing at 30+ FPS on mobile devices according to the official TensorFlow blog [9], making them ideal for real-time pose estimation in mobile applications. Therefore, the performance of both variations will be evaluated later on a mobile testing device.

2.2 2D Lightweight Pose Detection with BlazePose

Google ML Kit is an open-source mobile SDK that provides pretrained machine learning models for a variety of applications, allowing mobile application developers to easily integrate machine learning features into mobile applications. BlazePose [14] is the single-person PE model used in Google ML Kit, where it provides 33 2D keypoints on a human body that extends from the COCO format (see Figure 8).

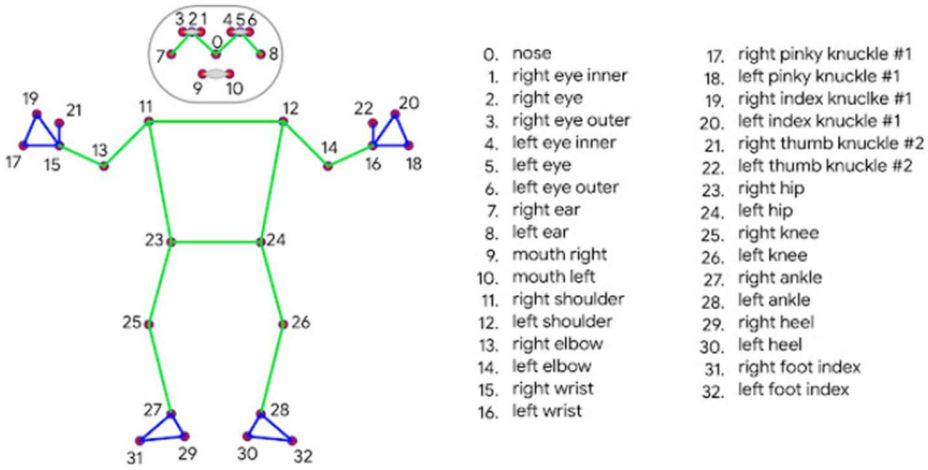


Figure 8. Keypoints output of BlazePose [14].

The single-person detection in person detector uses BlazeFace [15] face detection model as a proxy in BlazePose to predict additional information about person-specific alignment parameters, which helps locating the person within the frame. This is because Bazarevsky et al. [14] found the person's face gives the strongest signal to the neural network regarding the position of the torso due to its high-contrast features and a consistent structure. Therefore, by using BlazeFace face detector, BlazePose can accurately locate the position of target user for pose tracking later on (see Figure 9).

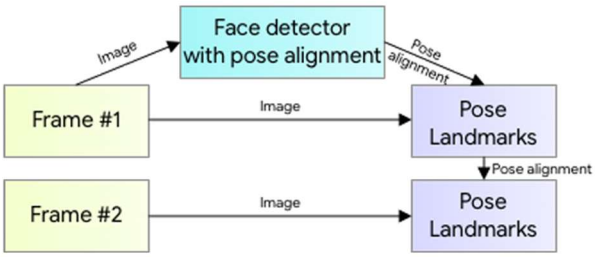


Figure 9. BlazePose inference pipeline [14]

Also, to make BlazeFace lightweight, it adapted depthwise separable convolutions that similar to MobileNet V2, which lower the computation load of facial detection by reducing the required parameters in the neural network significantly, results in a less complex model

structure will fewer layers (see Figure 10).

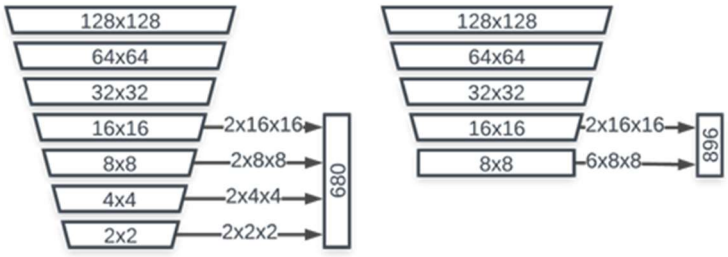


Figure 10. Anchor computation: SSD (left) vs. BlazeFace [14].

For pose tracking, the training of BlazePose has adapted the combined heatmap-offset and regression approach. In the heatmap-offset approach [16], heatmaps and their corresponding offsets are generated from training images for each of the 33 keypoints (see Figure 11), where the heatmaps indicate the Gaussian approximation of the keypoints lying in that specific region. Then, the heatmap and offset losses are computed and minimized during backpropagation in training, while a lightweight embedding is also learned to capture relevant features from the feedback of the heatmaps and offsets. Once the embedding is trained, a regression network is used to refine the predictions and make precise predictions of the keypoint positions, outputting them as 2D coordinates. Also, it is found that by skipping come connections between stages of network and applying gradient-stopping techniques significantly enhance both the accuracy of heatmap predictions and the precision of keypoints. (see Figure 12).

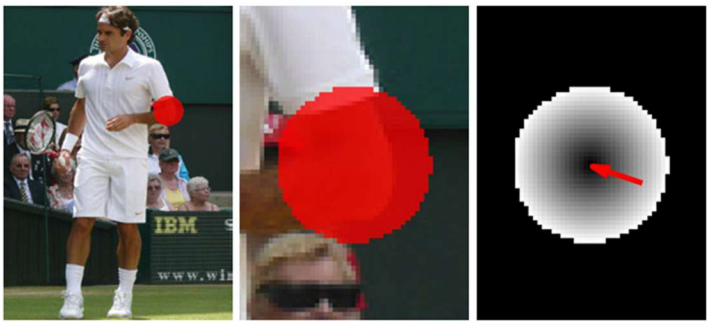


Figure 11. Heatmap target for the left-elbow keypoint (left & middle). Offset field L2 magnitude and 2D offset vector (Right) [16]

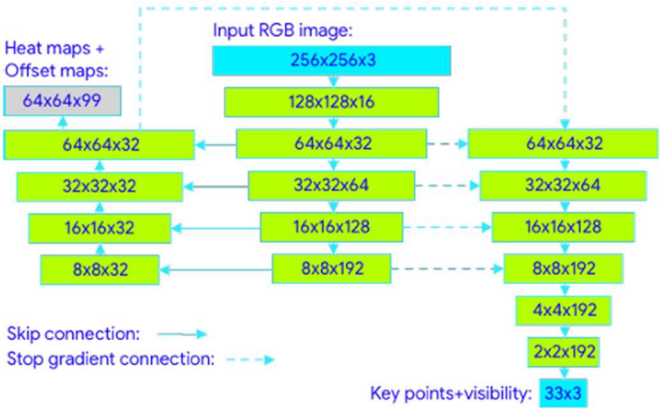


Figure 12. Network architecture of BlazePose [14]

To overcome occlusions, BlazePose also takes inspiration from Leonardo's Vitruvian man (see Figure 13), where parameters such as the midpoint of hips and radius of the human body can be predicted, maintain consistent performance in tracking human posture under complicated scenarios, such as overlapping limbs or significant occlusions [14].

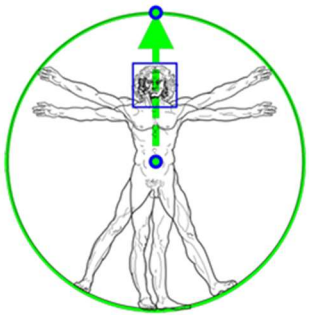


Figure 13. Leonardo's Vitruvian man [14].

Currently, there are two BlazePose models available for use: BlazePose Full with 3.5M parameters and BlazePose Lite with 2.7M parameters [14], both of which have demonstrated significant improvement on FPS while running on mobile devices compared to OpenPose, which is another lightweight PE model [20] (see Figure 14). Therefore, BlazePose Full is chosen as one of the potential PE models for this project, and its performance will be evaluated

later on.

Model	FPS	AR Dataset, PCK@0.2	Yoga Dataset, PCK@0.2
OpenPose (body only)	0.4	87.8	83.4
BlazePose Full	10	84.1	84.5
BlazePose Lite	31	79.6	77.6

Figure 14. BlazePose vs OpenPose [14].

2.3 Pose Estimation with State-of-The-Art YOLO model

Yolo11 [17] is the current state-of-the-art object detection model in the YOLO (You Only Look Once) family developed by Ultralytics, with outstanding performance comparing to the previous YOLO models (see Figure 15). Although there is not yet paper released regarding the details of YOLO11, the official blog has emphasized out several key enhancements comparing to the previous models.

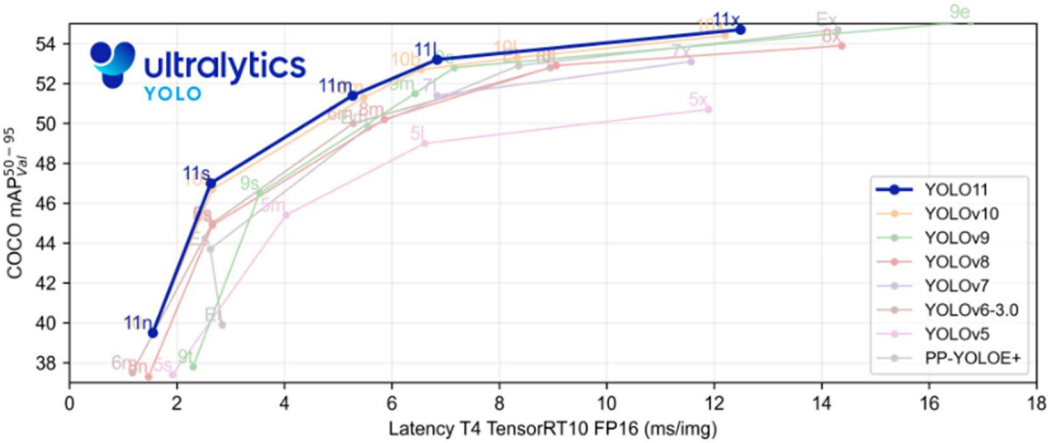


Figure 15. Object detection performance of various YOLO models [17]

Feature extractor being one of the improvements of YOLO11, has adapt an upgraded

backbone and neck architecture that allows enhancement of feature extraction capability, this makes enable object detection with higher accuracy compared to the previous models. Also, YOLO11's architecture and training pipeline has been further refined and optimized to enable a faster processing speed. With these improvements YOLO11 has achieved the highest mean average precision (mAP) on the COCO dataset compared all previous models while having 22% less parameters.

To achieve PE using YOLO11, the models is trained on the COCO keypoints dataset, with output being 17 2D keypoint coordinates and their corresponding confidence (see Figure 16). Currently there are 5 YOLO11 models with different scales and accuracies (see Figure 17), and YOLO11n-pose is chosen as one of the potential PE models used in this project due to its fewest number of parameters, making it more suitable for the limited computational power of mobile devices, and the performance will be evaluated and compared with other PE models.

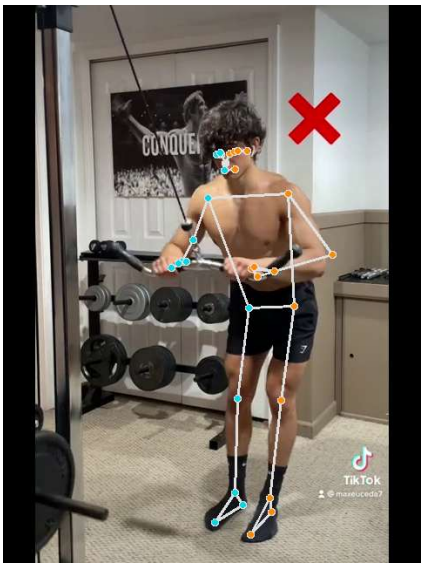


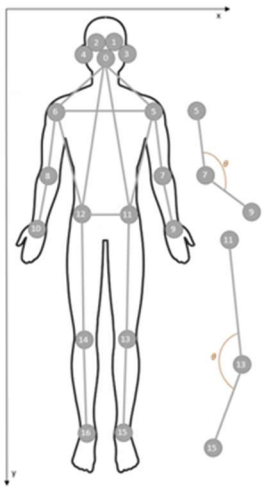
Figure 16. Keypoints outputted by YOLO11n-pose on an exercising frame from YouTube [18]

Model	size (pixels)	mAP^{pose} 50-95	mAP^{pose} 50	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n-pose	640	50.0	81.0	52.4 ± 0.5	1.7 ± 0.0	2.9	7.6
YOLO11s-pose	640	58.9	86.3	90.5 ± 0.6	2.6 ± 0.0	9.9	23.2
YOLO11m-pose	640	64.9	89.4	187.3 ± 0.8	4.9 ± 0.1	20.9	71.7
YOLO11l-pose	640	66.1	89.9	247.7 ± 1.1	6.4 ± 0.1	26.2	90.7
YOLO11x-pose	640	69.5	91.1	488.0 ± 13.9	12.1 ± 0.2	58.8	203.3

Figure 17. YOLO11 models with different scales [17]

2.4 Posture Monitoring with Joint Angle Calculation and YOLOv7

Kotte et al. [19] proposed a real-time posture correction approach for gym exercises using PE model, where they leverage the CNN model YOLOv7-pose [20] to identify 17 keypoints on the human body, then calculate joint angles using three connected keypoints for body movement analysis (see Figure 18). The user's posture is monitored by defining specific angle ranges for different joints, and computed angles that lie outside of these ranges are classified as incorrect posture. To support monitoring across a variety of gym exercises while reducing computational cost, only the keypoints crucial to each exercise are used to compute the joint angles (see Figure 19).



$$\phi(y,x) = \begin{cases} \arctan(\frac{y}{x}), & \text{if } x > 0, \\ \frac{\pi}{2} - \arctan(\frac{x}{y}), & \text{if } y > 0, \\ -\frac{\pi}{2} - \arctan(\frac{x}{y}), & \text{if } y < 0, \\ \arctan(\frac{y}{x}) \pm \pi, & \text{if } x < 0, \\ \text{undefined}, & \text{if } x = 0 \text{ and } y = 0. \end{cases}$$

$$\theta(u,v,p) = \frac{180(\phi(y_p - y_v, x_p - x_v) - \phi(y_u - y_v, x_u - x_v))}{\pi}$$

Figure 18. Keypoints output from YOLOv7 (left) and joint angle calculation using 3

connected keypoints (right) [19]

Exercises	Keypoints (Angle Ranges)											
	5	7	9	11	13	15	6	8	10	12	14	16
Pushups	[210°,280°]						[210°,280°]					
Bicep Curls	[010°,150°]						[010°,150°]					
Lunges				[142°,321°]						[153°,066°]		
Squats				[220°,280°]						[220°,280°]		
Shoulder Press	[026°,180°]						[026°,180°]					
Shoulder Lateral Raise	[171°,194°]						[171°,194°]					

Figure 19. Keypoint combination for selected sports exercises [19]

Kotte et al.'s approach provides valuable insights and demonstrations on utilizing PE technologies for posture monitoring in gym activities. By focusing only on the keypoints that directly contribute to the exercise, computational cost can be reduced because redundant keypoints are ignored during angle analysis, which is well-suited for mobile application development of this project. Additionally, they have employed a trigonometry approach that efficiently calculate joint angles, which also acts as an error classification method to perform performance analysis and provide feedback, this can potentially serve as the posture correctness classifier for this project. However, the experiment conducted by Kotte et al. was based on the YOLOV7-W6 [20] model with 70.4M parameters, making it unfeasible for mobile devices due to the high computation requirement. Therefore, other models with fewer parameters (e.g., BlazePose) will be used for implementing and testing this classification approach.

2.5 Posture Classification with Machine Learning Algorithms and OpenPose

OpenPose is a multi-person PE model developed by Cao et al. [21], capable of outputting keypoints in COCO, MPII, and COCO+Foot formats (see Figure 20).

(a) MPII (b) COCO (c) COCO+Foot

Figure 20. Keypoint annotation configuration for the 3 datasets [19]

Gupta et al. [22] have experimented using the keypoint output of OpenPose to train various machine learning algorithm for classification tasks. In the experiment, they used approximately 1000 images across 5 activity categories (sitting, standing, running, dancing, and laying) as input of PoseNet to obtain corresponding 2D keypoints, with labeling on each keypoint set corresponding to their own category. Then, these labeled keypoints sets are used to perform supervised learning on Logistic Regression, and Support Vector Machine (SVM) for activity classification. Unsupervised learning is also performed on K-Nearest Neighbor (KNN), Decision Tree, and Random Forest algorithm for such task. After training, these algorithms are tested with the testing dataset, and found Logistic Regression has the best performance while decision tree has the worst (see Figure 21).

Algorithms	Precision (%)	Recall (%)	F1-Measure (%)
Logistic	80.72	81.47	80.95
KNN	77.90	77.89	77.12
SVM	80.43	81.14	80.46
Decision Tree	74.49	75.80	73.50
Random Forest	80.75	80.34	79.43

Figure 21. Classification performance of 5 machine learning algorithms [22]

Gupta et al.'s experiment demonstrated satisfactory results using machine learning

algorithms to classify different postures based on keypoints outputted from a PE model. However, since it is yet to be determined whether machine learning algorithms will be used as the posture classifier in this project, further performance evaluation needs to be conducted on the testing mobile device alongside the PE model.

3. System Design and Solution

The purpose of this section is to provide a detailed overview of the project's implementation designs and logics.

3.1 Deliverables and Frame Classification Logic

When the user opens the application, they are given a selection of weight training exercises. When an exercise is selected, the exercise information including the image demonstration, muscles involved, step-by-step text guide, YouTube video demonstration link, and information source link will be displayed with a “start exercise” button.

After the “start exercise” button is clicked, the front camera will activate, and posture monitoring begin. Each frame captured by the front camera will feed to the PE model to detect the positions of various body joints as 2D coordinates. These coordinates will then be passed to a classifier that determines the correctness of posture in each frame. When the posture in a frame is classified as “wrong posture”, the mobile phone will continuously to vibrate, ring, and display a red boarder to alert the user, until a correct posture is classified (see figure 22).

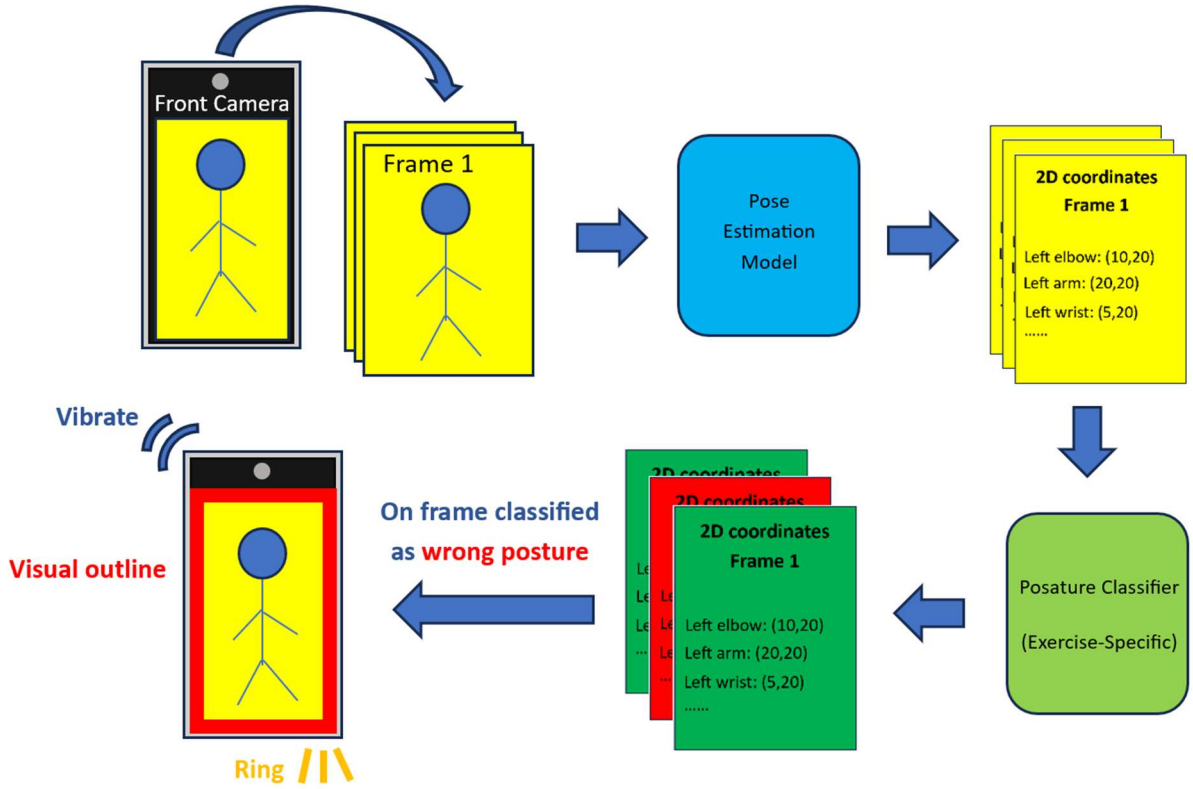


Figure 22. Frame classification and feedback process.

Lastly, when user double taps the screen, a general posture correctness results will be provided based on the number of inaccurate posture frames. Also, user will be able to see the history records and video playbacks by saving the records and entering the history page.

3.2 Devices for Application Development and Testing

The development of the mobile app is done on Android Studio using programming language Kotlin, and the performance testing involved a physical Android testing device with Android API 28, and two emulated devices with Android API 32 and 35 (see Figure 23). The the mobile applications are packed and built into the devices for functionality testing, but only the physical testing device is used for performance testing.

Device Name	Samsung Galaxy S8 (2017)	Pixel 6 (emulator)	Small Phone (emulator)
Processor	Qualcomm Snapdragon 835	N/A	N/A
Android API	API 28	API 32	API 35
Ram	4 GB	N/A	N/A

Figure 23. Specifications of testing devices.

3.3 System Diagrams and User Interface Design

For the detailed system designs of major functions, including use case, sequence, and class diagrams, see the following table:

Diagrams & figures	Appendix
Use case diagram	See Appendix A
Class diagrams	See Appendix B
Role of each class	See Appendix B (Figure B8)
Sequence diagrams	See Appendix C
User Interface for each page	See Appendix D

Figure 24. UML diagrams and their corresponding appendix references.

3.4 Logic Flow: Feedback System

The feedback system provides users with feedback when a workout frame is classified. Currently, there are three main types of feedback: visual, vibration and sound feedback.

For visual feedback, a thick border is displayed around the screen, and the color of the border changes to green for each frame classified as a correct posture and red for an incorrect posture.

For vibration and sound feedback, the mobile phone vibrates or rings whenever a frame is classified as an incorrect posture and stops when a frame is classified as a correct posture. Also, users can activate or disactivate the vibration and ringing feature by pressing a button on the Selection Screen (see Figure 25).

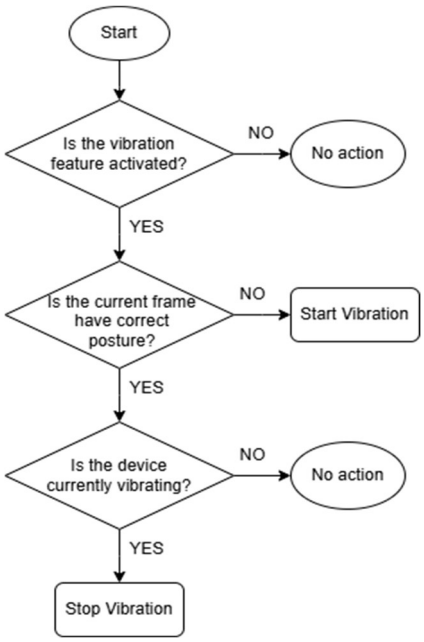


Figure 25. Logic flow of vibration feedback (same logic flow applied to sound feedback).

3.5 System Design: Plug-In Based Content Extension Design

To enhance the extendibility of this project and allow new weight training exercises can be added with ease, a plug-in-based content extension design has been introduced to minimize the modification of existing code.

When a new weight training exercise is added to the project, a folder including the corresponding classifier class (exercise specific classification rules) and a display function (information desired to be displayed during monitoring process) is required to be inserted into the *ExercisePlugIn* folder. Then, the classifier list in *ClassifierRegistry* file and display list in *DisplayClassifierFeatures* file need to be updated to include the newly added classifier class and display function for system recognition. After the lists are updated, the system is able to recognize the extension and update the corresponding UI, and perform monitoring with desired rules when the new exercise is selected by user.

For content display of new exercise in *PrePoseStartScreen* file, an asset folder that contains text (in JSON format) and image (in JPG format) content is required to be added in the asset folder for automatic retrieval when *PrePoseStartScreen* file is loaded (see Figure 26).

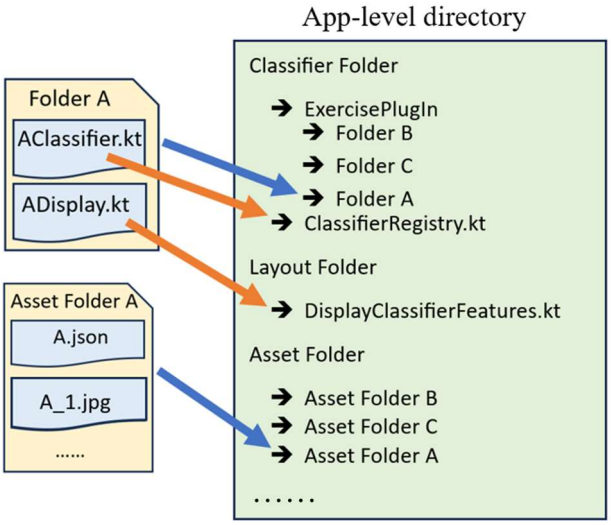


Figure 26. Files and folders involved in Exercise A extension, including file-adding actions (blue arrows) and file-modification actions (orange arrows).

4. Component Benchmarking and Selection

4.1 PE Model Benchmarking Overview

There are five PE models selected for this project: BlazePose Full, BlazePose Lite, MoveNet Thunder, MoveNet Lightning, and YOLO11n-pose. Since only one PE model will be utilized and implemented in this project, the PE models underwent benchmarking across different phases as part of the model selection process. In each benchmarking phase, only models with satisfactory performance will be considered for implementation and will proceed to the next evaluation stage.

There are 2 benchmarking evaluation stages (see Figure 27). The first evaluation focuses on the FPS of the PE models while running on the testing device, which aims to explore usability of different PE models and ensure a smooth user experience when using the application in real-time. The second evaluation focuses on the accuracy of the qualified PE models, where the detected keypoints of the human body are compared to one another with respect to ground truth data to ensure the reliability of the detection results. Such design of evaluation allows desired PE model to be selected effectively while avoiding the redundant benchmarking on undesired PE models.

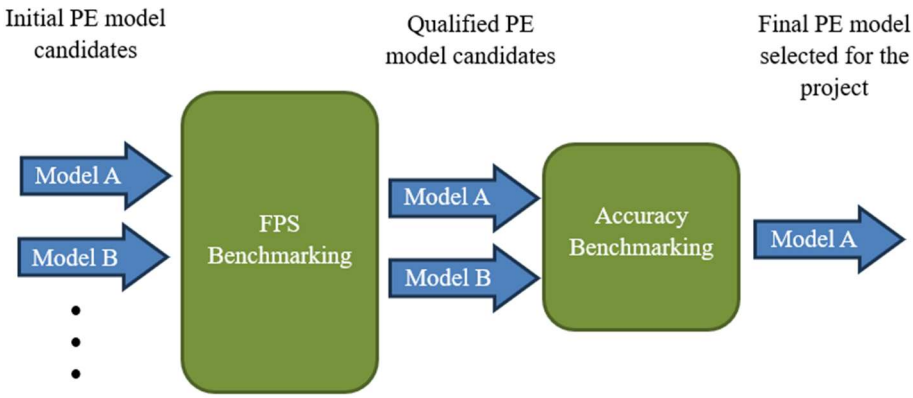


Figure 27. PE model selection pipeline.

4.2.1 PE Model Benchmarking: FPS

The frames per second (FPS) of a PE model is defined as the rate at which the model processes input frames and outputs the detected human keypoints in real-time every second, using a continuous input stream from the front-facing camera of a mobile device.

As the main objective of this project is to develop a user-friendly mobile application capable of monitoring a user's posture in real time, balancing the portability and usability of the PE models is considered crucial in the model selection process. According to the research done by Chen & Thropp [23], tasks involving psychomotor and perceptual performance are best performed at frame rates above 15 Hz, where lower frame rates (5 Hz – 10 Hz) significantly impair task performance and are perceived as lower in quality. Also, in terms of watchability and satisfaction. Liu et al. [24] also discovered that videos presented at 5 Hz frame rates are "unacceptable" to viewers, as they lead to lower visual quality, whereas videos presented at 15 Hz and above are generally more widely acceptable by the viewers. Therefore, ensuring that the output frame rates of the PE models are sufficient to maintain smooth keypoint display and rendering can lead to a more satisfactory user experience.

Therefore, during the FPS benchmarking phase, each PE model is run on the testing device for 5 minutes, with the frame rate recorded every second, and only models with results above 10 FPS are considered satisfactory. To achieve such monitoring, a fully functional Android application is built for each PE model, installed on the testing device, and configured to log the frame rate to the Android Studio console every second. To ensure fairness during the FPS monitoring process, each application consists of only three classes: *MainActivity* for the application's page navigation and camera input logic, *PoseAnalyzer* for extracting keypoints output from the PE model, and *OverlayView* for rendering and sketching the keypoints and skeleton overlay on input frames for display.

Based on the FPS monitoring results of the five PE models (see Figure 28), the qualified models include BlazePose Lite with an average of 21.9 FPS, BlazePose Full with an average of 16.87 FPS, and MoveNet Lightning with an average of 12.14 FPS, suggesting that these models require less to less computational resources and thus they are better suited for real-time mobile application. However, YOLO11n-pose and MoveNet Thunder performed poorly, with averages of 0.88 FPS and 3.22 FPS respectively, which fell below the FPS threshold and highlighted their intensive computational demand and unsuitability for real-time mobile application.

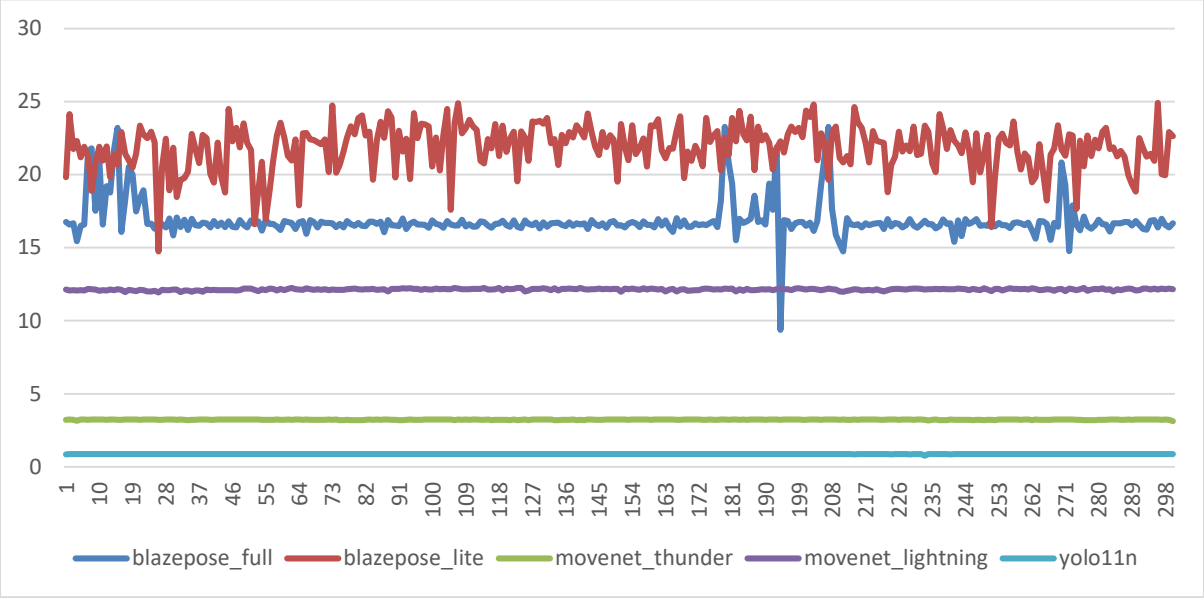


Figure 28. 5-minute FPS comparison of 5 PE models on the testing device (Frames/Second).

4.2.2 PE Model Benchmarking: Accuracy

For the accuracy benchmarking, a set of 10 images was selected from the COCO annotations_trainval2017 dataset [25], where each image features a person in different postures (exercising, standing, sitting) with clearly visible body parts or with occlusions (see Figure 29). Such selection provides a more general view of model accuracy under different scenarios, thereby providing a more comprehensive evaluation of the models' performance.

Figure 29. Example illustrating an image with clearly visible limbs (left) and limb occlusion (right), with keypoints are highlighted using ground truth annotations.

To measure the accuracy of the qualified PE models, only the 12 common output keypoints by BlazePose Full, BlazePose Lite, and MoveNet Lightning are considered (LEFT_SHOULDER, RIGHT_SHOULDER, LEFT_ELBOW, RIGHT_ELBOW, LEFT_WRIST, RIGHT_WRIST, LEFT_HIP, RIGHT_HIP, LEFT_KNEE, RIGHT_KNEE, LEFT_ANKLE, RIGHT_ANKLE). The models are first used to predict the human joint keypoints in from of normalized 2D coordinates (values scaled to [0,1] by default, then multiplied by 100 for more interpretable or comparable results) for the selected image set. The predicted keypoints are then compared to the corresponding ground truth annotations provided in the dataset. The difference between the predicted and true keypoints is measured using the L2 norm, and the final accuracy for an image is computed by the mean squared error (MSE) of these keypoint differences (see Formula 1). Lastly, the individual MSE values are averaged across the 10 images for each model, and the averaged MSE are served as the accuracy performance of the PE models, with lower values indicating lower error and higher accuracy.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(100 \times (\hat{x}_i - x_i))^2 + (100 \times (\hat{y}_i - y_i))^2}$$

Formula 1. MSE calculation for each selected image.

Figure 30 illustrates the accuracy benchmarking results of the 3 models. BlazePose Full achieved the best performance with an average MSE of 1.172, outperforming BlazePose Lite (average MSE of 2.33) and MoveNet Lightning (average MSE of 23.70). By observation, there are significant fluctuations in MSE are found in MoveNet Lightning, and by comparing the highest MSE with the lowest, the value fluctuates for more than 6440% in specific images, which accounts for the large differences of MSE compared to the other two PE models. However, even when excluding these high MSE cases, BlazePose Full still outperformed the other two models with an average MSE of 1.06, compared to BlazePose Lite with 2.26 and MoveNet Lightning with 1.99.

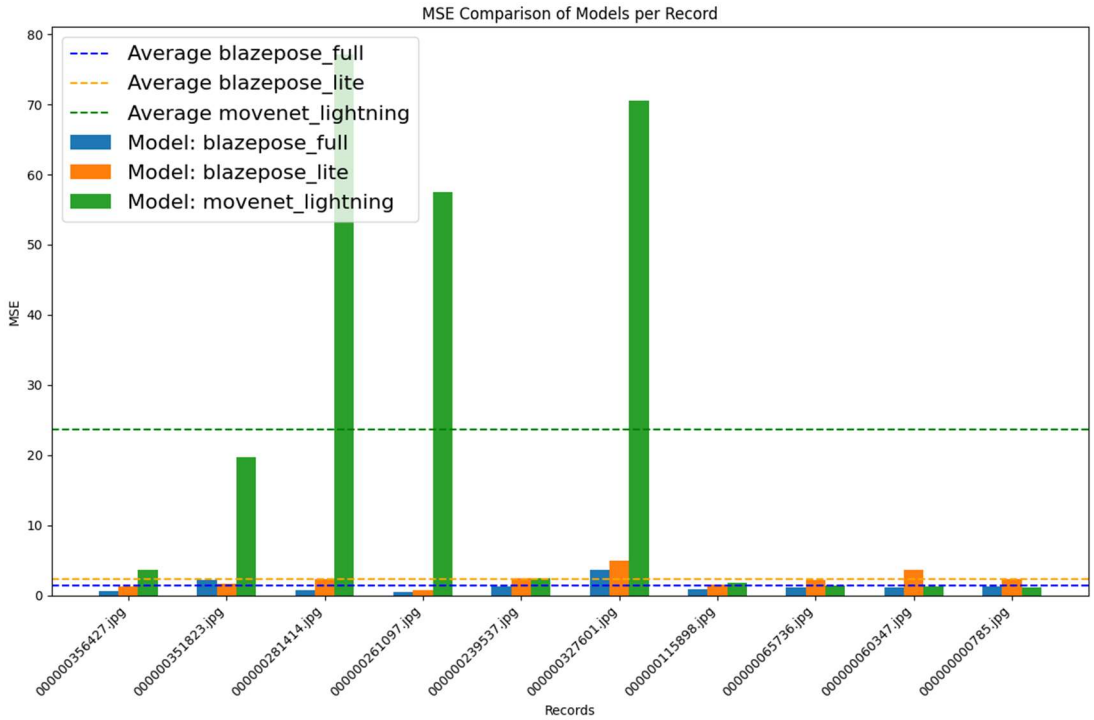


Figure 30. MSE comparison of BlazePose Full, BlazePose Lite and MoveNet Lightning on each selected COCO image.

To investigate the cause of the MSE fluctuations observed in MoveNet Lightning, the

keypoints detected by the models were sketched and compared against the ground truth annotations of the evaluation images. For example, there exists leg occlusion for the person in the image 351823, MoveNet Lightning struggled to differentiate the leg keypoints correctly, which results in a mix-up of left and right leg detections, while BlazePose models correctly pinpointed the keypoints under occlusions (see Figure 31). Similar results can be found in other images where MoveNet Lightning struggled to accurately distinguish left from right, or it even missed keypoints entirely when limbs are appeared to be occluded in the images (see Figure 32).

Figure 31. Comparison of left (red) and right (blue) body part outlines between the PE model outputs and ground truth.

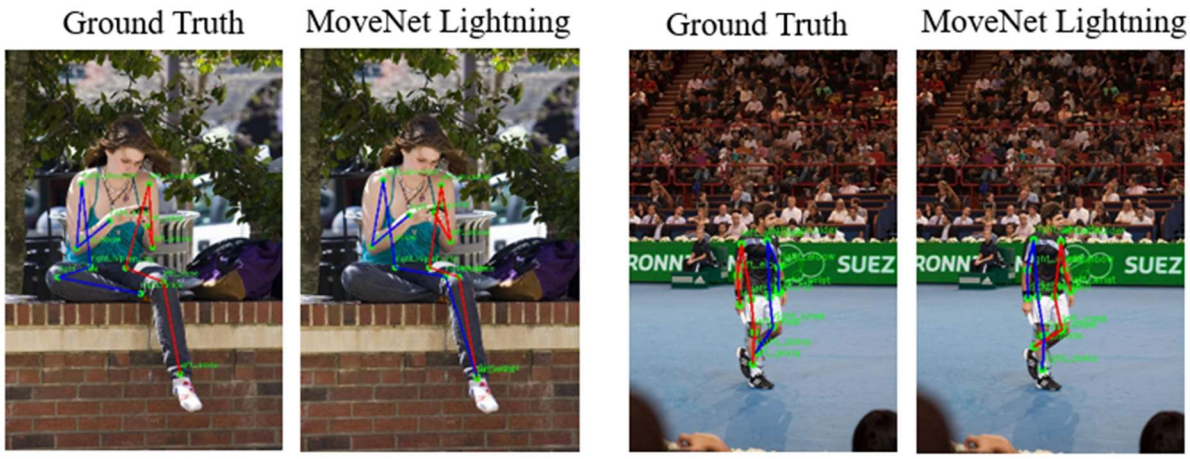


Figure 32. Example of keypoints detection failure (left) and left-right swap error (right) in

MoveNet Lightning.

The accuracy benchmark results have demonstrated the importance of overcoming occlusion in PE tasks, especially in gym activities where occlusions are inevitable due to complex body movement and the use of gym equipment during weight training. BlazePose models, inspired by Leonardo da Vinci's Vitruvian Man [14], are specifically trained to address these occlusions. Such specialized training has demonstrated the noticeable improvement in keypoint detection accuracy under different human poses compared to MoveNet, which lacks such specialized training, making BlazePose models more suitable for more suitable for this project.

Therefore, BlazePose Full has been selected for this project due to its outstanding accuracy while being able to maintain a satisfactory FPS comparing to the other PE models.

4.2 Classifier Benchmarking

The posture classifier plays an important role in this project, where it is responsible for receiving the keypotins from the PE model, and determine the correctness of the user, therefore the classifier is required to have a satisfactory accuracy in order to provide reliable posture feedback to users.

The classifier approaches that been tested are the angle calculation approach and machine learning algorithms, which include Random Forest, SVM, KNN and Logistic Regression.

4.2.1 Classifier Benchmarking

Evaluating the accuracy of classifying weight training exercise has been found to be challenging due to the difficulty in collecting a dataset of posture correctness in weight training

activities that is sufficiently large for training ML algorithms and achieving a balanced distribution of correct and incorrect postures. Therefore, inspired by human activity recognition tasks, the classification of sitting and standing postures is used to simulate the classification of posture correctness in weight training activities, where sitting is classified as a “wrong posture”, while standing is classified as a “correct posture”.

The North-West University has organized a dataset containing the normalized keypoints outputted by an OpenPose PE mode [26]. In this project, 10,000 records (5,000 sitting, 5,000 standing) are selected and used for the classifier benchmarking, where 80% of the dataset will be used for training while 20% of the dataset will be used for accuracy validation.

During the classifier benchmarking, only the keypoints involved in the motion of sitting (LEFT_SHOULDER, LEFT_HIP, LEFT_KNEE, RIGHT_SHOULDER, RIGHT_HIP, RIGHT_KNEE) were used from the dataset, which avoids the computation on redundant keypoints and improve motion specification (see Figure 33).

Figure 33. The keypoints used for classifying standing (left) and sitting (right) posture, outlined by shoulder, hip and knee keypoints.

For angle calculation approach, the decision boundary between standing and sitting postures is set at 135°, representing the midpoint between posture of standing completely straight (180°) and a completely seated (90°). To quantify the posture, the angle formed at the hip is computed using the coordinates of shoulder (x_1, y_1), hip (x_2, y_2), and knee (x_3, y_3), and the angle is derived using the cosine rule of vectors (see Formula 2).

$$\theta = \cos^{-1} \frac{(x_2 - x_1)(x_3 - x_2) + (y_2 - y_1)(y_3 - y_2)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \times \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2}}$$

Formula 2. Angle calculation formula using the cosine rule for three 2D coordinates.

For the machine learning algorithms, each record in the dataset is transformed into a feature vector with 12 dimensions, including the x and y coordinates of keypoints of shoulder, hip, and knee. These features are then passed to the algorithms as input, allowing them to learn patterns based on the spatial relationships between these keypoints (see Figure 34).

Approach	Learning Type	Settings
K-Means Clustering	Unsupervised Learning	• K = 2 • Centroids: Random initialization
SVM	Supervised Learning	• Kernal: linear
Logistic Regression	Supervised Learning	• Max iteration: 1000 • Regularization: L2
Gaussian Naïve Bayes	Supervised Learning	N/A
Random Forest	Supervised Learning	• Number of Estimators: 100 trees

Angle Computation	N/A	• Classify as sitting: <135° • Classify as standing: >135°

Figure 34. Classifier benchmarking settings.

4.2.2 Classifier Benchmarking: Results & Conclusion

The classifier benchmarking results of all approaches are shown in Figure 35. For ML algorithms, SVM, Random Forest and Logistic Regression have achieved a satisfactory classification accuracy of 94%, 93% and 94%, whereas K-Means Clustering and Naïve Bayes performed poorly with only 67% and 48.3% accuracy. For angle calculation approach, it has obtained an 87.1% accuracy.

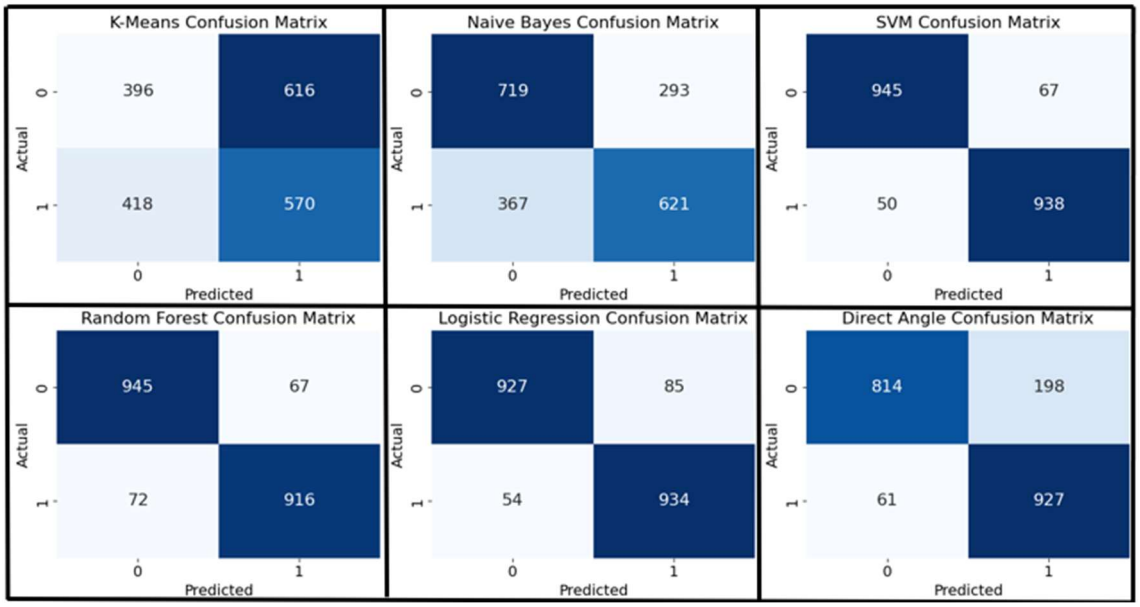


Figure 35. Confusion matrix of various classifiers.

To understand the learning patterns of the ML algorithms, training sets with different sizes were used to perform supervised learning (see Figure 36), where no obvious accuracy-increasing trend was found, suggesting that they had reached their optimal performance. By

further zooming into the first 100 training sets used, SVM, Random Forest, and Logistic Regression began to demonstrate the ability to progressively capture the spatial relationships between the keypoints and learn from the training data (see Figure 37).

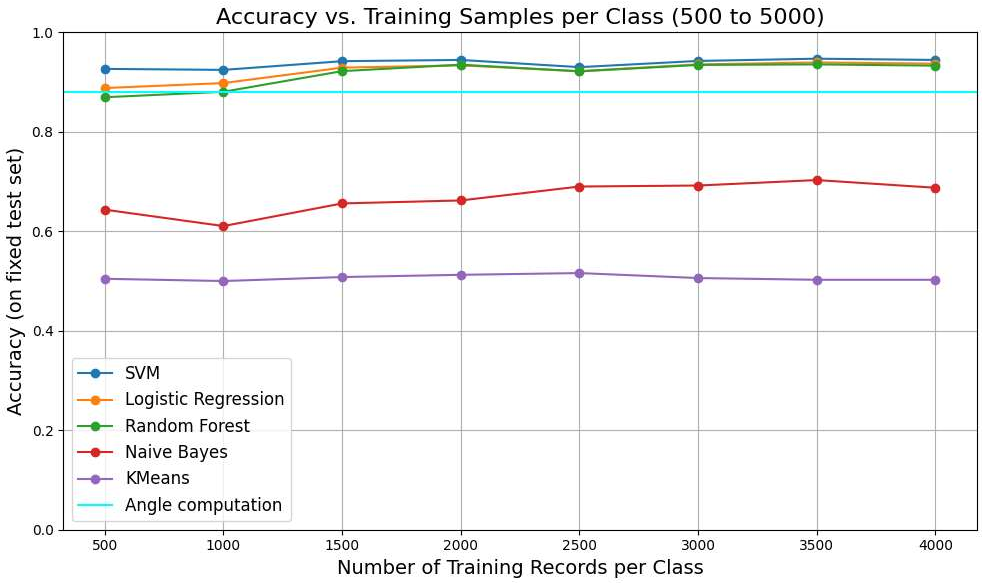


Figure 36. Classification accuracy vs training samples (scaled by 500 records).

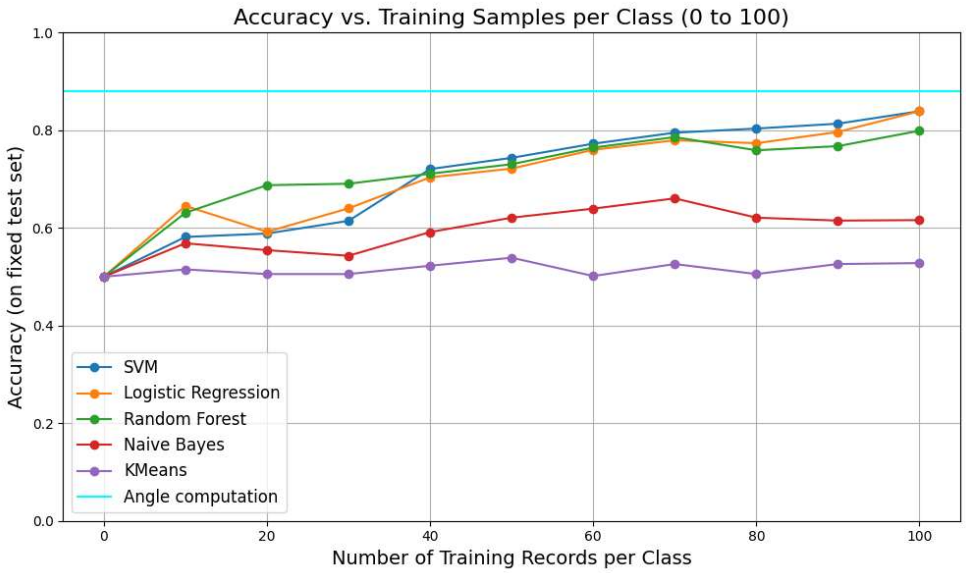


Figure 37. Classification accuracy vs training samples (scaled by 10 records).

In contrast, Bayes Naïve and K-means clustering struggled to capture such patterns. For Bayes Naïve, the feature vector violated the assumption of feature independence, as the positions of the shoulders, hips, and knees are relatively dependent on each other. For K-means clustering, the algorithm heavily depends on the initial selection of centroids, which can lead to suboptimal clustering results if the centroids are not well-chosen.

The benchmarking results have demonstrated that the direct angle classification approach is more efficient and suitable for this project in various scenarios. In cases where there is a sufficient amount of data to train different classifiers, classifying postures using computed angles from keypoints has shown to be relatively accurate (87.1% accuracy), although it performed slightly worse than Logistic Regression, SVM, and Random Forest. However, in scenarios where there is limited training data, machine learning algorithms struggle to provide functional classification results, while the angle classification approach can maintain high classification accuracy.

If ML algorithms are implemented as classifier in this project, introducing a new weight training activity for posture monitoring will require the training and data gathering of a new ML classifier, which makes the machine learning approach less scalable compared to the angle classification approach. But when classifying posture using computed angles, a decision boundary based on the posture definition be easily defined for classification without the need for extensive retraining. Therefore, such scalability and simplicity make the angle calculation approach a more practical solution for this project, especially in situations with limited data or when more exercises are added to this project.

5. Posture Monitoring Feature Implementation

5.1 Classification Setups

From the previous section, it was concluded that performing posture classification is scalable while providing relatively accurate results when using 2D keypoints detected from the PE model as input. To define a uniform standard for posture classification, only angles that are multiples of 15° are used as classification boundaries for simplicity. Also, the mean of computed angles for left and right body part will be used to avoid the errors caused by positional asymmetry.

5.2 Classification Tasks

There are two main classification tasks that provide posture monitoring features: repetition counting and posture monitoring.

For the posture monitoring task, the classifier receives the keypoints detected by the PE model and classify rather the posture in the current frame is correct. Here, only the angles of the joints that significantly contribute to the correctness of posture during weight training activities are computed. These computed angles must lie within a predefined range to classify the frame as correct, and angles fall outside the defined range are classified as incorrect (see Figure 38).

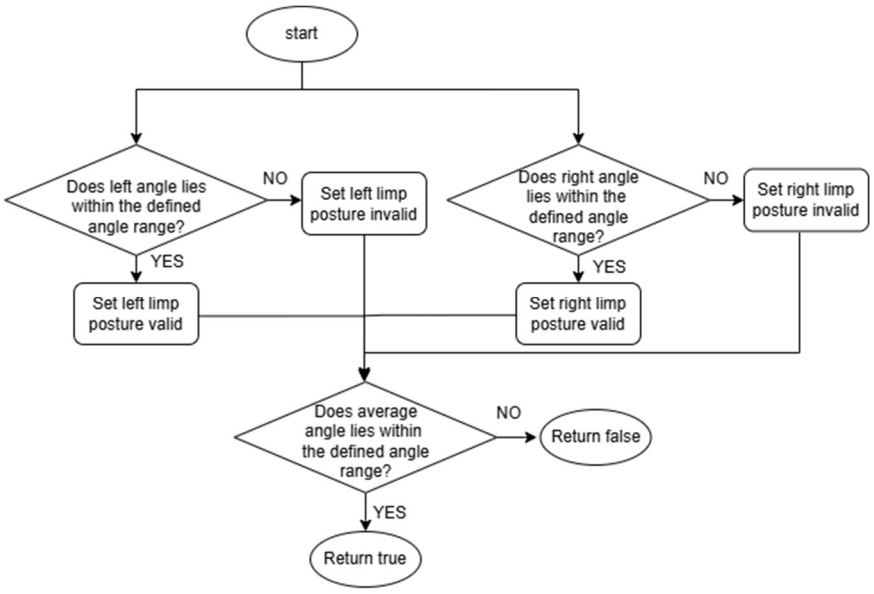


Figure 38. Logic flow for posture monitoring. A return true indicates the posture in current frame is classified as correct, while a return false indicates the posture in current frame is classified as incorrect.

For the repetition counting task, which is similar to the posture monitoring task, received the keypoints and determine rather a repetition is completed. Here, only the joints that contribute to the motion of the limbs are used for classification. The computed angles must increase above or decrease below certain thresholds to classify the motion as either "up motion" or "down motion," depending on the direction of the movement (see Figure 39).

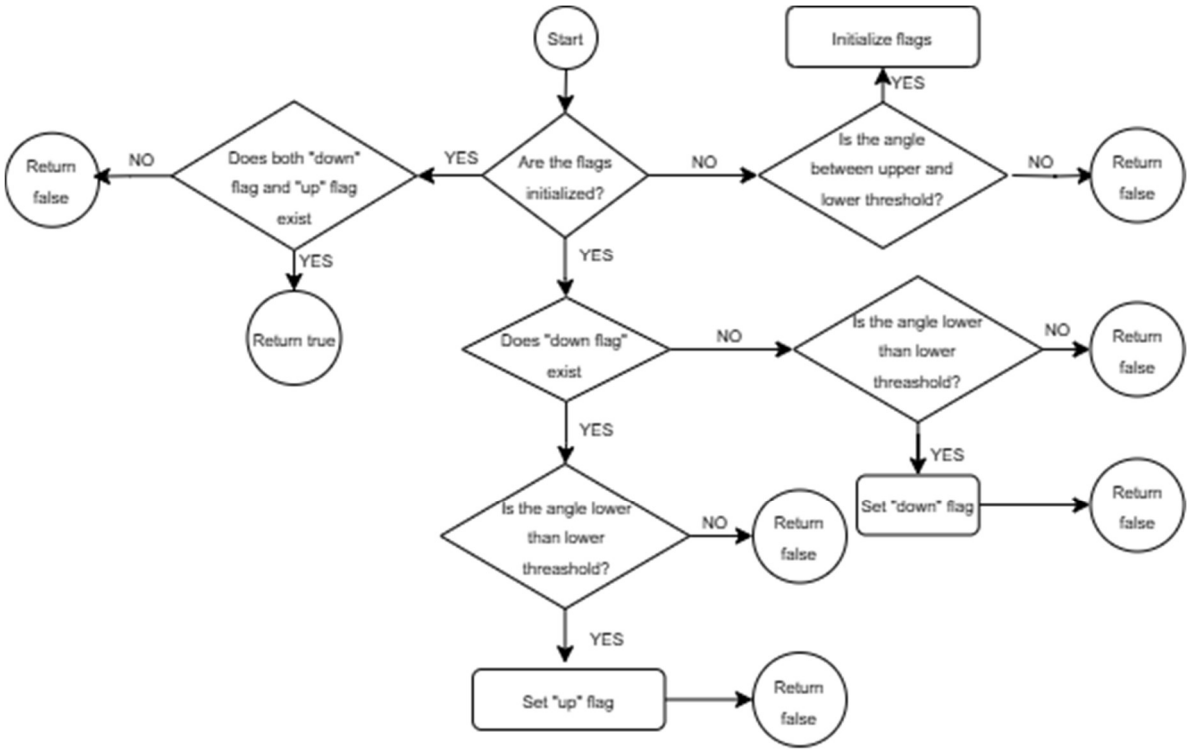


Figure 39. Logic flow for repetition counting. A return true indicates the completion of one repetition, while a return false indicates either an invalid input or that the repetition is still in progress.

5.3 Implemented Exercises and Specifications

The information provided to the users in *PrePoseEstimationSreen* and angles defined for classification tasks for each exercise are obtained from Training.fit. Training.fit is a publicly accessible platform that aims to share reliable workout instructions and tips for weight training beginners, and they claim the provided information is based on the academic backgrounds of Training.fit members and scientific publications [27].

The posture monitoring objectives for each exercise are defined based on the "Common Mistakes and Injuries" and "Video Tutorial" sections of the exercise page on Training.fit. Then, appropriate angles for classification tasks are selected according to the objectives and the "Step-by-step instructions" section of the same exercise page (refer to Appendix D for exercise-

specific objectives and specifications in classification tasks).

5.4 Real-life Posture Monitoring Demonstration

For the detailed real-life posture monitoring demonstration images for each implemented weight training exercises, please refer to the following table for their corresponding appendix:

Exercise	Appendix
Straight arm pulldown	F1
High Cable Curls	F2
Cable triceps pulldown	F3
Front squad	F4
Barbell Row	F5
Incline Chest Press	F6
Dumbbell shoulder press	F7
Sit-up	F8
Push-up	F9

Figure 40. Real-life posture monitoring demonstrations with their corresponding appendix references.

6. System and Component Functionality Testing

Application's functionality tests on components/functions were performed on different Android systems to evaluate their compatibility in terms of obtaining system permissions, ensuring proper functionality, and complying with changing security policies. For the testing result of each function, a "pass" indicates the function has meet the expectation, whereas a "fail"

indicates the function does not meet the expectation or leads to serious errors such as app crash (refer to Appendix E for the complete testing results of major components/functionalities).

7. Conclusion

This section provides a self-review of the development process of the project, including major achievements, encountered problems, and limitations.

7.1 Major Achievements

One of the major achievements of this project is to study and understand the capability of various CNNs in image processing and PE tasks. To gain relevant knowledge and prepare for this project, the architectures of different CNN models were studied, enabling an understanding of how each model approaches PE tasks, such as the involvement of face detection for localization, the use of depthwise separable convolutions for the reduction of computational intensity, and the utilization of heatmaps and offsets to refine predicted keypoints (refer to Section 2).

Additionally, the project explored the possibility of utilizing PE models for portable use in the weight training domain, which resulted in a functional Android application with real-time image analysis and classification features for monitoring workout postures. To select a suitable PE model for portable application, the FPS and keypoint detection accuracy of some existing lightweight PE models are compared by implementing them into Android applications and executing them on a physical testing device. This enabled a direct comparison of the computation intensiveness and highlighted different shortcomings between these PE models. By integrating classification logics in exercise-specific classifiers with selected PE model, analysis and classification of exercise posture on incoming frames can be done locally with minimal delays, resulting in real-time posture monitoring feature.

7.2 Major Problem Encountered

The first major problem encountered was finding existing weight training exercise datasets for classifiers' training and comparison. To overcome this challenge, inspiration was drawn from human activity recognition tasks, where datasets containing 'sitting' and 'standing' activities are used to train the classifiers and simulate the classification of 'correct posture' and 'wrong posture' (refer to Section 4.2.1). This solution ensured that the dataset used for training the classifiers was sufficient with evenly distributed labels, allowing the results to accurately reflect the performance of different classification approaches.

The second major issue encountered was the incompatibility of the developed app with newer Android systems, which was discovered during the application functionality testing phase. This led to the failure of the screen recording feature and caused the app to crash (refer to Appendix E). To debug this issue, breakpoints were set at various locations in the source code, and the logged errors were analyzed. It was found that the cause of the issue was related to changes in security policies regarding screen recording in Android API Level 30 (Android 10) and higher [28]. By studying the security policies and adjusting the source code for policy compliance, the app was able to run without errors, and the screen recording feature functioned correctly. This problem also highlighted the importance of complying with OS-specific policies during application development, which is crucial in the software development life cycle.

7.3 Limitation and Potential Improvement: PE Model

BlazePose Full although stands out among other lightweight pose estimation models in terms of usability and accuracy, it struggles to perform keypoint detection when the human face is not clearly visible in the processed image. This limitation is caused by the architecture of the model, where BlazePose uses the BlazeFace face detection model to locate the human face and then performs keypoint detection on the remaining body parts [14]. While this

approach enhances keypoint detection accuracy and processing speed for lightweight applications, detection completely fails when BlazeFace is unable to locate the human face in the image, as the detection of body keypoints relies on the detected face location. In contrast, models like YOLO11x-pose [17] that are not depend on this feature, are able to detect keypoints even when the face is not visible or clearly shown (see Figure 41).

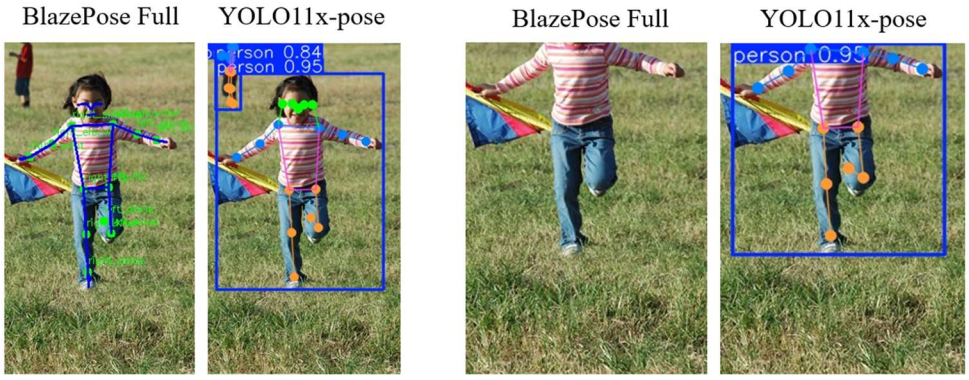


Figure 41. Keypoint output comparison between BlazePose (face detection involved) and YOLO11x-pose (face detection not involved).

However, it is common that during weight training sessions, users experience limited space due to crowded or compact gym environments, which may make it difficult for them to position themselves so that their face is visible to the front camera. As a result, BlazePose may fail to accurately capture their posture in such case, as the model relies on face detection to locate keypoints.

To overcome this limitation, more powerful pose estimation models that are trained directly on human pose data can be used in this project, but these models may exceed the computational capabilities of portable devices.

7.4 Limitation and Potential Improvement: Hardware

With the rapid advancement of technology, newer mobile phones are released each year that have featured more powerful processors. By directly comparing the 3DMark gameplay physics simulation results of the testing device (Samsung Galaxy S8) with a one of the latest mobile phones, it is obvious that there is a major leap in CPU performance over the years, reflecting the advancements in processing technology and the increasing demand for higher performance to support AI applications and gaming (see Figure 42). Also, with the raise popularity of AI applications, more recent mobile phones have equipped with powerful processors optimized for running various AI models (e.g. Snapdragon 8 Elite AI Engine) [30], resulting in newer mobile phones being capable of running more heavy weighted models locally with lower latency.

Processor (Year)	Device	3DMarks Benchmarking score	Processing Unit (Frequency)	AI Features
Snapdragon 835 (2017)	Samsung Galaxy S8	2390	4 x Kryo 280 CPU cores (Up to 2.45 GHz each)	N/A
Snapdragon 8 Elite (2024)	Asus ROG Phone 9	16497	2 x Prime Qualcomm Oryon CPU cores (up to 4.47 GHz) 6 x Performance Qualcomm Oryon CPU cores (up to 3.53 GHz)	Qualcomm AI Engine

Figure 42. Hardware comparison of the testing device of this project (2017) [29] and the newly released mobile device (2024) [30]

Therefore, using an older mobile phone as a testing device in this project, the benchmarking results may not accurately reflect the performance capabilities of the newest mobile phone models available in the current market, leading to selection of the PE model that is more suitable for older mobile phone models. Whereas using the newer mobile phone, it may provide the selection of more resource-intensive PE models that could offer higher keypoint detection accuracy while still maintaining high FPS due to the advanced computational capabilities provided by the latest generation of processors.

8. Schedules

Task	Start Date	Estimated End Day	Duration
Perform literature reviews on gym injuries and existing PE solutions	10th Sept, 2024	27th Oct, 2024	1.5 month
Design the application architecture and overall logics	15th Oct, 2024	28th Nov, 2024	1.5 month
Perform performance testing on PE models on mobile device	29th Nov, 2024	15th Dec, 2024	2 weeks
Perform performance testing on different posture classification approaches	16th Dec, 2024	31th Dec, 2024	2 weeks
Develop and implement the mobile application (Functions & UI)	1st Jan, 2025	1th Mar, 2025	2 months
Perform overall testing and issue resolution, perform final	2th Mar, 2025	31th Mar, 2025	1.5 month

documentation			

9. Monthly Logs

Month	Log
September, 2024	• Perform literature reviews on gym injuries • Explore existing PE models and their applications • Complete Project Plan
October, 2024	• Initial design of application architecture and overall logics • Select potential PE models and posture classifying approaches for this project • Complete Interim Report I
November, 2024	• Designed & confirmed the overall system design and interactions • Built functional mobile application using YOLO11, MoveNet, and BlazePose for performance benchmarking, with key points being able to overlay on the correct body parts of human in captured video frames in real-time. Lines are also sketched to connect the key points to present a skeletal form of captured human posture.
December, 2024	• Building functional mobile application using YOLO11, MoveNet, and BlazePose for performance benchmarking. • Designing & preparing performance benchmarking (e.g. PE performance, posture classifier performance).
January, 2025	• Perform PE model performance experiment • Perform posture classifier experiment • Application development

February, 2025	• Application development: completed page navigation logic • Application development: implemented plug-in-based design for content extension • Application development: completed history record saving and loading logic • Application development: implemented dynamic UI designs (auto-ui-adjustment based on available plug-ins) • Application development: completed feedback system (visual, vibration, sound feedback) • Application development: resolved screen recording permission and compatibility issues across different Android API levels and systems • Application development: resolving video playback compatibility issues across different Andoird API levels and systems (currently supports Android API level 32 or lower, compatibility issues do not affect other features but video playback when app is running on Android API level 33 or higher)
March, 2025	• Application development: Added 9 exercises for posture monitoring (sit-up, push-up, high cable curls, straight arm pulldown, cable triceps pulldown, incline chest press, dumbbell shoulder press, barbell row, and front squad) • Documentation: Complete final report, demo video, and presentation slides

10. References

1. F. Hysi and Dervishi, "The relation between self-esteem and body dissatisfaction," British Journal of Psychology Research, vol. 12, 2024. [Online]. Available: https://eajournals.org/bjpr/vol12-issue-1-2024/the-relation-between-self-esteem-and-body-dissatisfaction/.

2. W. Thompson, "Worldwide survey of fitness trends for 2023," ACSM's Health & Fitness Journal, vol. 27, no. 1, 2023. [Online]. Available: https://journals.lww.com/acsm-healthfitness/fulltext/2023/01000/worldwide_survey_of_fitness_trends_for_2023.6.aspx.

3. M. Cuthbertson-Moon, P. A. Hume, H. E. Wyatt, I. Carlson, and B. Hastings, "Gym and fitness injuries amongst those aged 16-64 in New Zealand: Analysis of ten years of Accident Compensation Corporation injury claim data," Sports Med Open, vol. 10, no. 1, p. 53, May 2024. doi: 10.1186/s40798-024-00694-9. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/38744758/.

4. H. A. Bukhary, N. A. Basha, A. A. Dobel, R. M. Alsufyani, R. A. Alotaibi, and S. H. Almadani, "Prevalence and pattern of injuries across the weight-training sports," Cureus, vol. 15, no. 11, p. e49759, Nov. 2023. doi: 10.7759/cureus.49759. [Online]. Available: https://www.cureus.com/articles/49759.

5. BeOne Sports, *BeOne Sports*, [Mobile App]. BeOne, 2023. [Online]. Available: https://apps.apple.com/ua/app/beone-sports/id1666571774.

6. Y. Zengin, "Fitness Trainer Pose Estimation," GitHub, 2021. [Online]. Available:

https://github.com/yakupzengin/fitness-trainer-pose-estimation.

7. H. Novatchkov and A. Baca, "Artificial intelligence in sports on the example of weight training," J. Sports Sci. Med., vol. 12, no. 1, pp. 27–37, Mar. 2013. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC3761781/.

8. A. K, P. Prabu, and J. Paulose, "Human Body Pose Estimation and Applications," in *2021 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 2021, pp. 1-6. doi: 10.1109/i-PACT52855.2021.9696513. [Online]. Available: https://ieeexplore.ieee.org/document/9696513.

9. TensorFlow, "MoveNet: Ultra fast and accurate pose detection model," TensorFlow Hub. [Online]. Available: https://www.tensorflow.org/hub/tutorials/movenet?hl=zh-cn.

10. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," arXiv, 2019. [Online]. Available: https://arxiv.org/abs/1801.04381.

11. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv, 2017. [Online]. Available: https://arxiv.org/abs/1704.04861.

12. T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," arXiv, 2017. [Online]. Available: https://arxiv.org/abs/1612.03144.

13. X. Zhou, D. Wang, and P. Krähenbühl, "Objects as Points," arXiv, 2019. [Online].

Available: https://arxiv.org/abs/1904.07850.

14. V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "BlazePose: On-device Real-time Body Pose tracking," arXiv, 2020. [Online]. Available: https://arxiv.org/abs/2006.10204.

15. V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs," arXiv, 2019. [Online]. Available: https://arxiv.org/abs/1907.05047.

16. G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy, "Towards Accurate Multi-person Pose Estimation in the Wild," arXiv, 2017. [Online]. Available: https://arxiv.org/abs/1701.01779.

17. Ultralytics, "YOLOv11 Models Documentation," Ultralytics Documentation. [Online]. Available: https://docs.ultralytics.com/models/yolo11/.

18. MaxEuceda7, "NUr8_anD1nM," YouTube, Shorts video. [Online]. Available: https://www.youtube.com/shorts/NUr8_anD1nM.

19. H. Kotte, M. Kravčík, and N. Duong-Trung, "Real-Time Posture Correction in Gym Exercises: A Computer Vision-Based Approach for Performance Analysis, Error Classification, and Feedback," ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/publication/374659509_Real-Time_Posture_Correction_in_Gym_Exercises_A_Computer_Vision-Based_Approach_for_Performance_Analysis_Error_Classification_and_Feedback.

20. C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," arXiv, 2022. [Online].

Available: https://arxiv.org/abs/2207.02696.

21. Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," arXiv, 2019. [Online]. Available: https://arxiv.org/abs/1812.08008.

22. A. Gupta, K. Gupta, K. N. M. Gupta, and K. O. Gupta, "Human activity recognition using pose estimation and machine learning algorithm," in *International Symposium on Intelligent Control*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:232401456.

23. J. Y. C. Chen and J. E. Thropp, "Review of Low Frame Rate Effects on Human Performance," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 37, no. 6, pp. 1063-1076, Nov. 2007, doi: 10.1109/TSMCA.2007.904779.

24. R. T. Apteker, J. A. Fisher, V. S. Kisimov, and H. Neishlos, "Video acceptability and frame rate," *IEEE MultiMedia*, vol. 2, no. 3, pp. 32-40, 1995, doi: 10.1109/93.410510.

25. Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015). Microsoft COCO: Common Objects in Context. *arXiv preprint arXiv:1405.0312*. Available: https://arxiv.org/abs/1405.0312.

26. J. du Toit, “Human pose dataset (sit & stand pose classes)”. North-West University, 15-Jun-2023 [Online]. Available: https://dayta.nwu.ac.za/articles/dataset/Human_pose_dataset_sit_stand_pose_classes_/23290937/1.

27. Training.fit, "The right exercise just a click away," [Online]. Available: https://training.fit/.

28. Android Open Source. "Restricted Screen Reading." *Android Open Source*. [Online]. Available: https://source.android.com/docs/core/permissions/restricted-screen-reading.

29. Qualcomm, "Snapdragon 835 Product Brief," Qualcomm, [Online]. Available: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/snapdragon_product_brief_835_0.pdf.

30. Qualcomm, "Snapdragon 8 Elite Mobile Platform Product Brief," Qualcomm, 87-83196-1 REV C, [Online]. Available: https://docs.qualcomm.com/bundle/publicresource/87-83196-1_REV_C_Snapdragon_8_Elite_Mobile_Platform_Product_Brief.pdf.

11. Appendix

Appendix A

Use case Diagrams

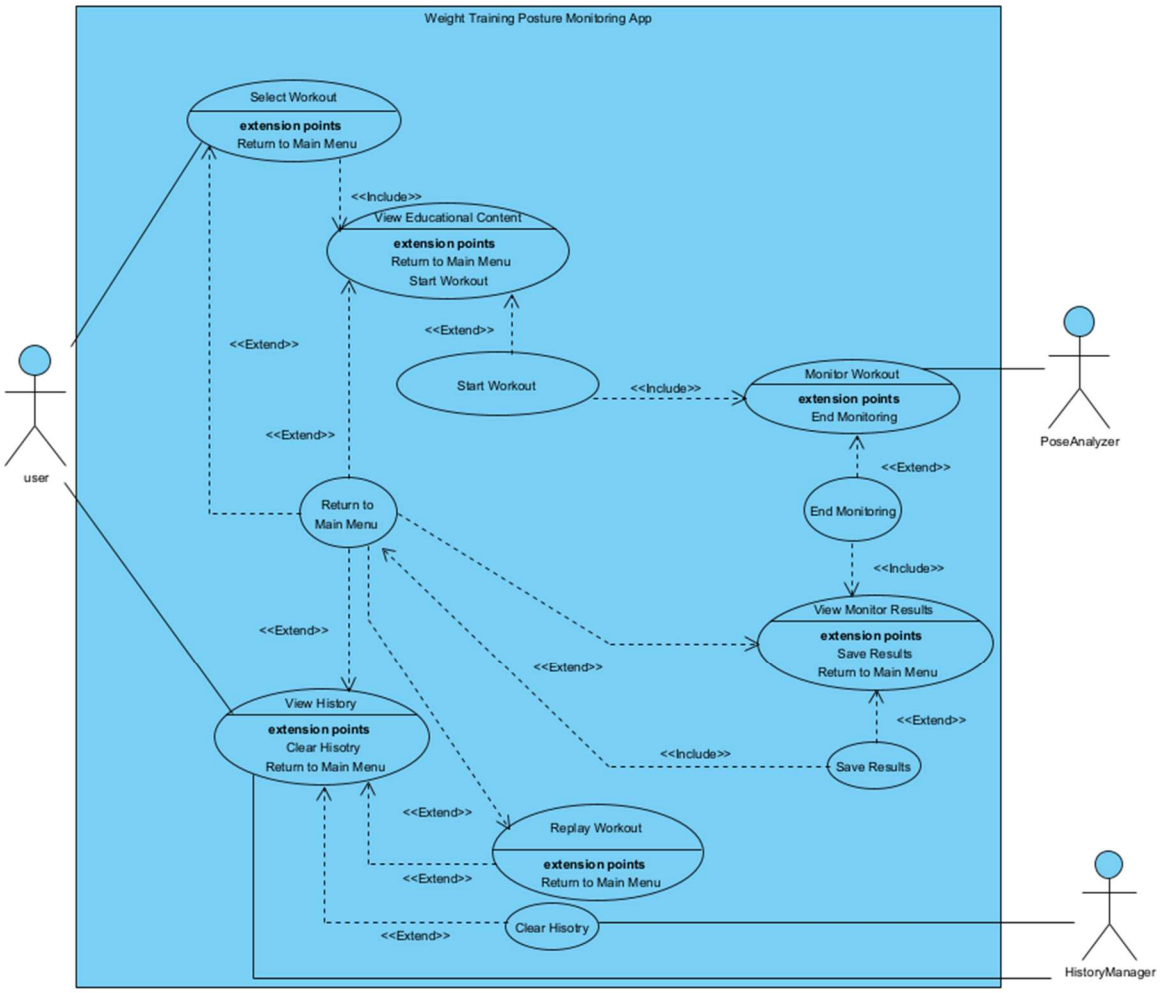


Figure A1. Use case diagram of the posture monitoring application

Appendix B

Class Diagrams

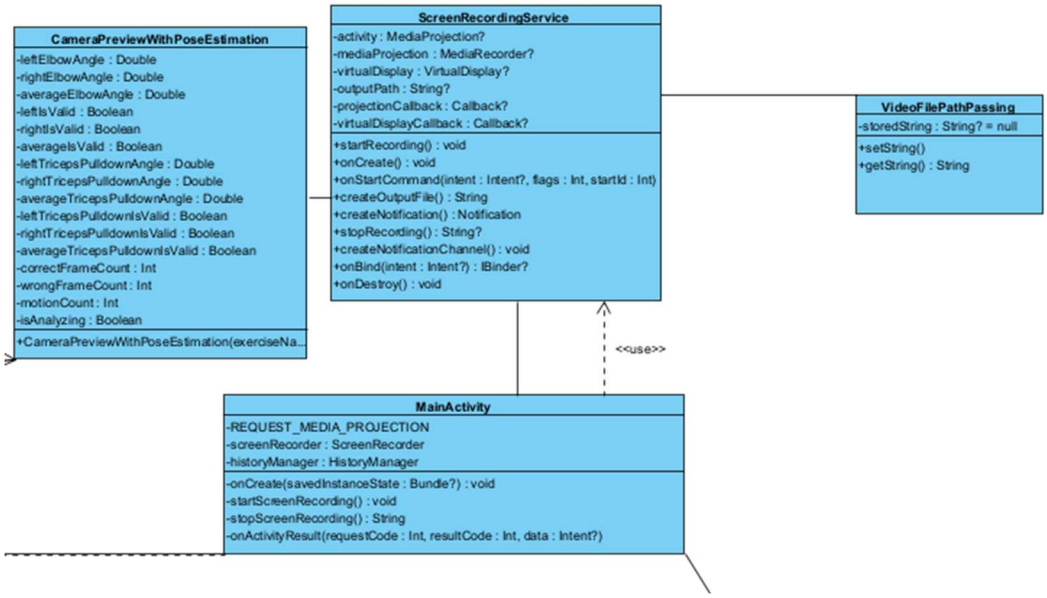


Figure B1. Class diagram: screen recording.

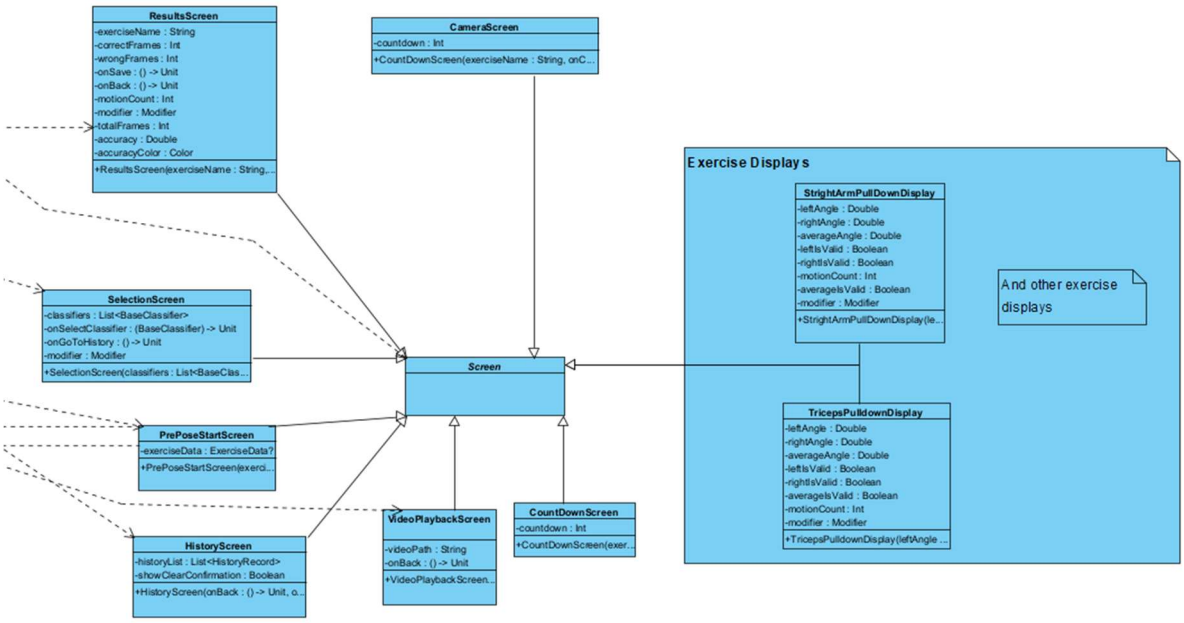


Figure B2. Class diagram: UI and display screens.

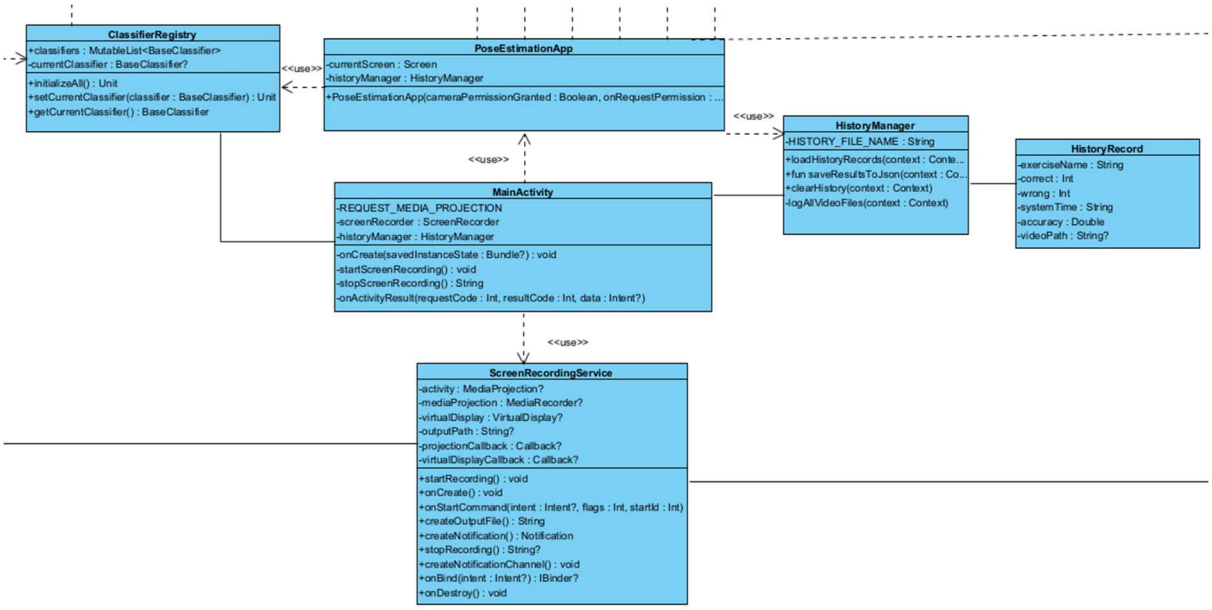


Figure B3. Class diagram: MainActivity and record loading/saving.

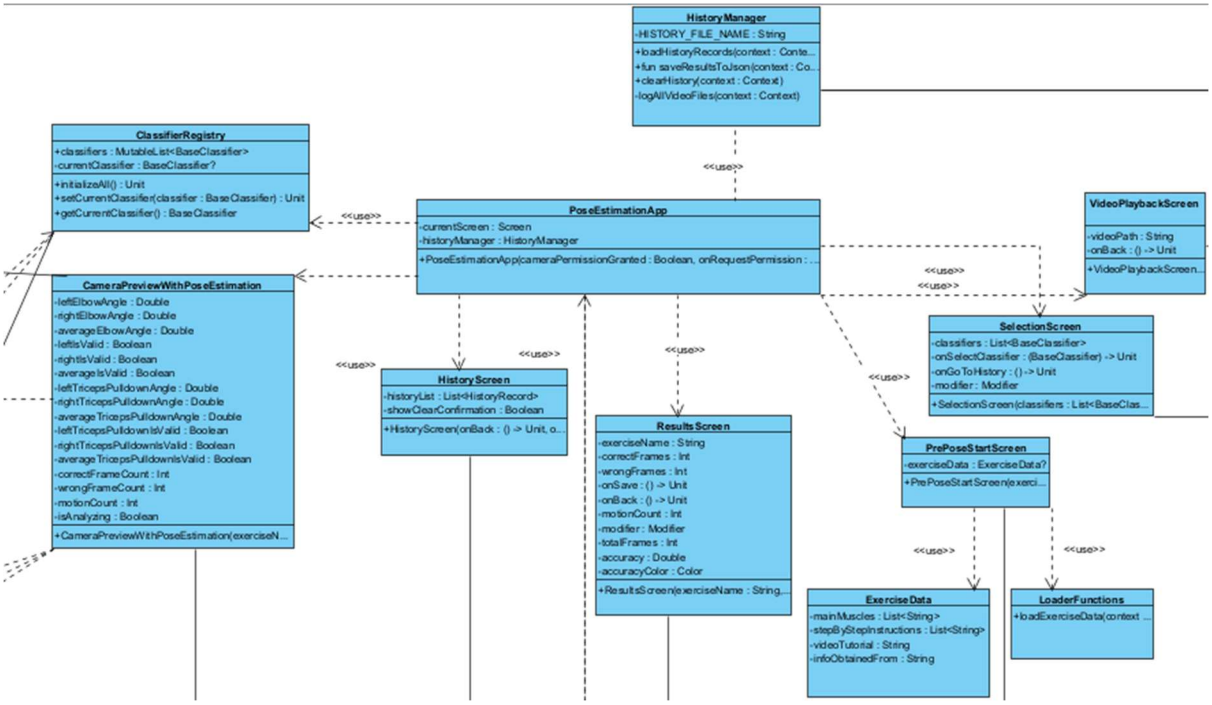


Figure B4. Class diagram: page navigations and information loading.

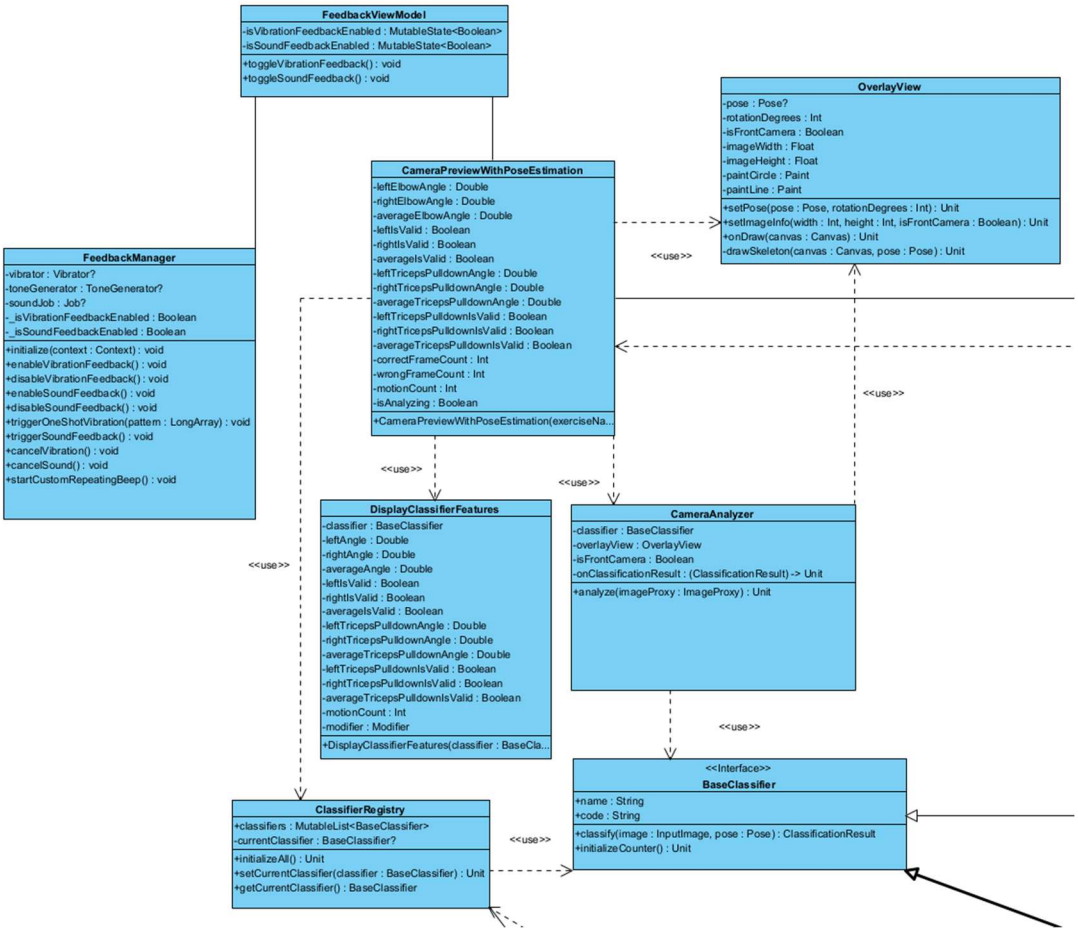


Figure B5. Class diagram: exercise-specific real-time posture monitoring with feature displays.

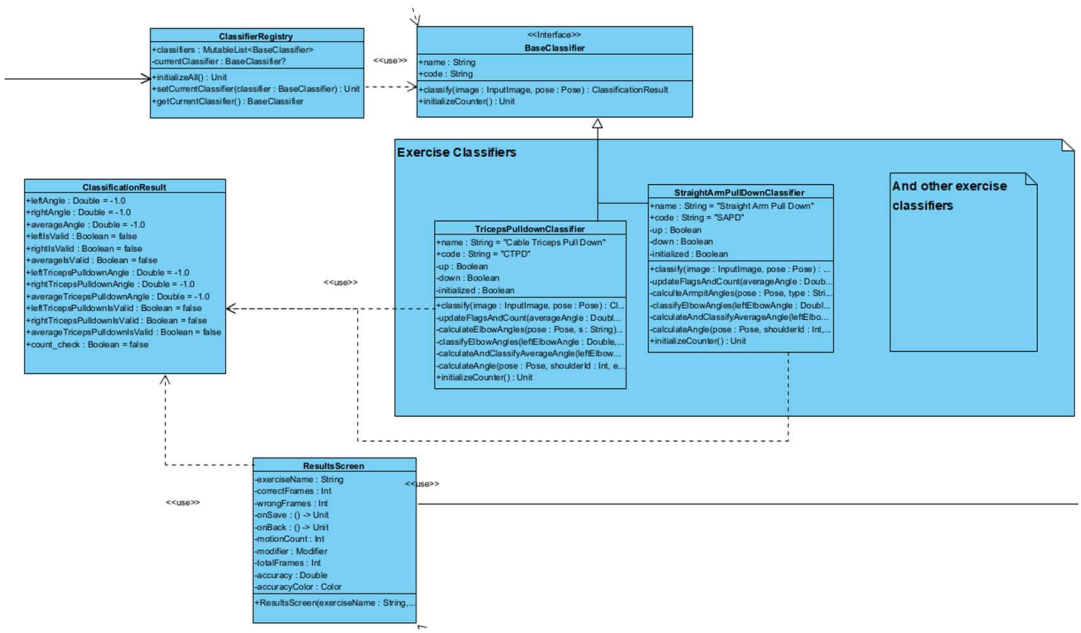


Figure B6. Class diagram: exercise-specific classifiers and results display.

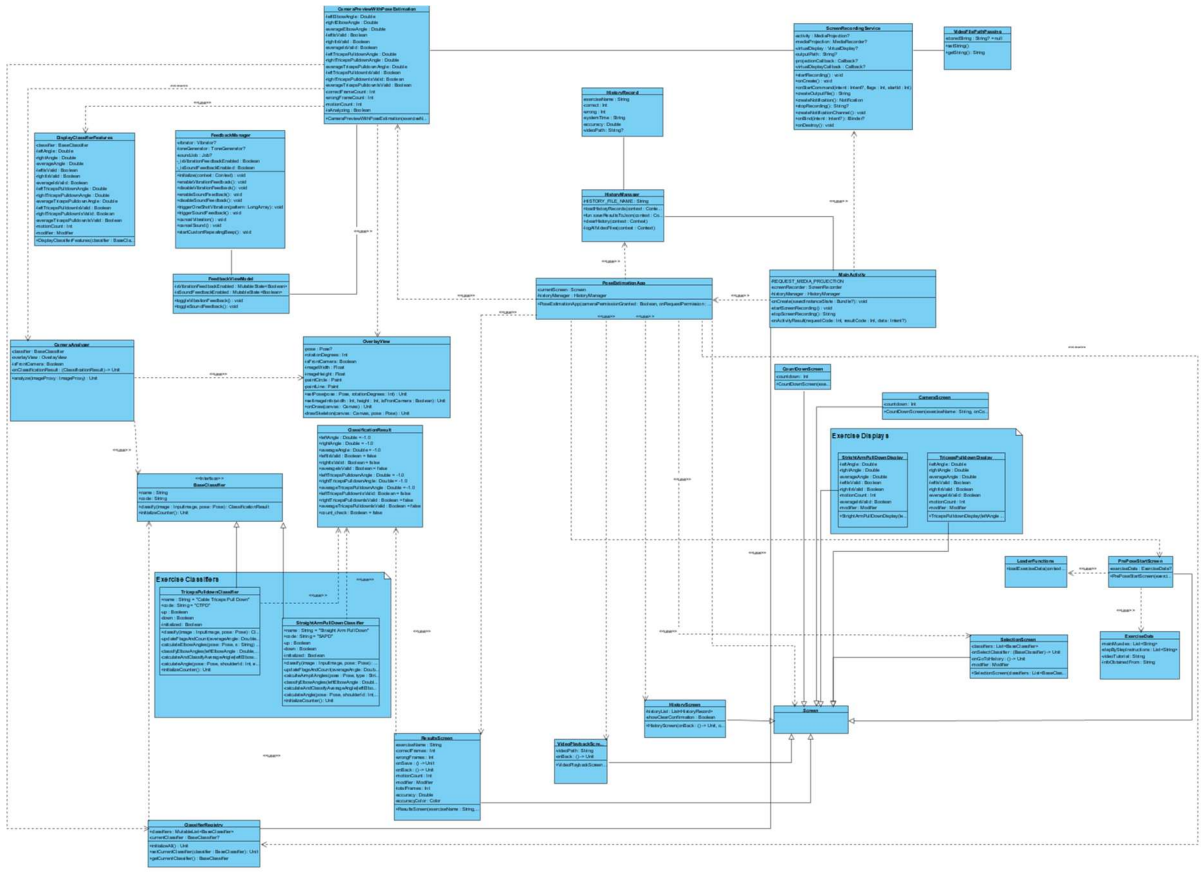


Figure B7. Class diagram overview

Class	Roles
MainActivity	Manages the app lifecycle, initializes classifiers, handles permissions, and screen recording.
PoseEstimationApp	Manages the navigation and user interface for the pose estimation application
ScreenRecordingServices	Handles screen recording functionality using MediaProjection and MediaRecorder. It also resolves compatibility issues across multiple Android API levels due to the change of security policy.
OverlayView	A custom view that visually overlays keypoints and skeletal connections of detected keypoints on the camera preview.
HistoryManager	Manages storing, retrieving, and clearing user session history, including exercise results and associated video recordings.
CameraAnalyzer	Processes camera frames for pose detection, updates the overlay view with detected poses, and classifies poses using the provided classifier, invoking a callback with classification results.
Screen	Sealed class, manages different navigation pages.
CameraPreviewWithPoseEstimation	Displays the camera preview and performs pose estimation, providing motion and accuracy results to the parent component.
SelectionScreen	Displays a list of exercise classifiers for users to select and provides navigation to the history screen.

PrePoseStartScreen	Displays exercise details, illustrations, and instructions, and provides options to start the exercise or navigate back.
CountDownScreen	Provides a countdown timer before starting the exercise and allows the user to cancel or proceed.
ExerciseDisplays	Displays angles, validation statuses, and motion count for different exercises.
StraightArmPullDownClassifier	Classification of posture correctness given input of 2D coordinates.
ResultsScreen	Displays the workout results, including accuracy, motion count, and exercise name, with options to save or navigate back.
HistoryItem (in HistoryScreen)	Represents an individual record in the history list with exercise details and accuracy percentage, providing a clickable interface for further actions.
HistoryScreen	Displays a list of exercise history records with options to view or delete them.
HistoryItem (in HistoryScreen)	Represents an individual record in the history list with exercise details and accuracy percentage, providing a clickable interface for further actions.
VideoPlaybackScreen	Provides a video playback interface with a media controller and a back button for navigation.
HistoryRecord	Represents a single history record with exercise details, performance metrics, and an optional video path.
ExerciseData	Represents detailed information about an exercise, including primary muscles involved, step-by-step

	instructions, a video tutorial link, and the source of the information.
BaseClassifier	Defines a common interface for classifiers, including methods for classifying poses and initializing counters, along with properties for the classifier's name and code.
ClassificationResult	Represents the results of a classification process, including angles, validity checks, and motion count status for pose-based exercises.
ClassifierRegistry	Manages a list of available classifiers and provides methods to initialize, set, and retrieve the current classifier.
ExerciseClassifiers	Implements exercise-specific logic for detecting and counting motion cycles during a workout session by analyzing different joint angles from pose data.
VideoFilePathPassing	Passes the file name of screen recordings to other classes for video file saving.
FeedbackManager	Handles vibration feedback and sound feedback when wrong posture is classified.
FeedbackViewModel	Monitors the status of vibration feedback and sound feedback.
InfoScreen	Display the tutorial of the application.

Figure B8. Classes and roles.

Appendix C

Sequence Diagrams

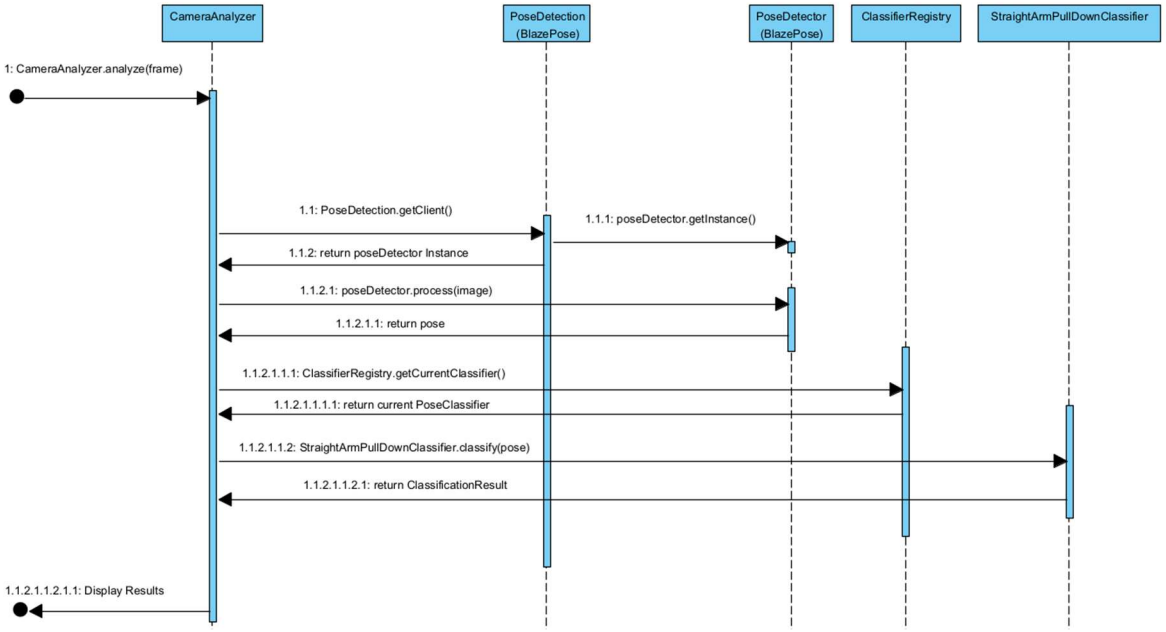


Figure C1. Sequence diagram: frame classification process.

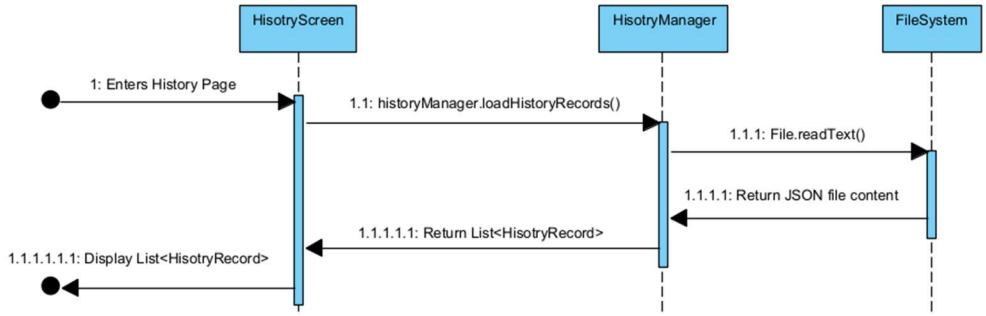


Figure C2. Sequence diagram: loading history records.

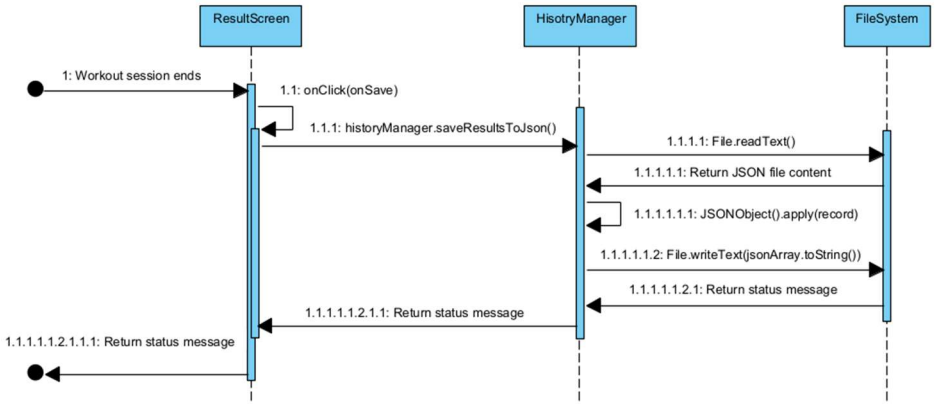


Figure C3. Sequence diagram: saving history records.

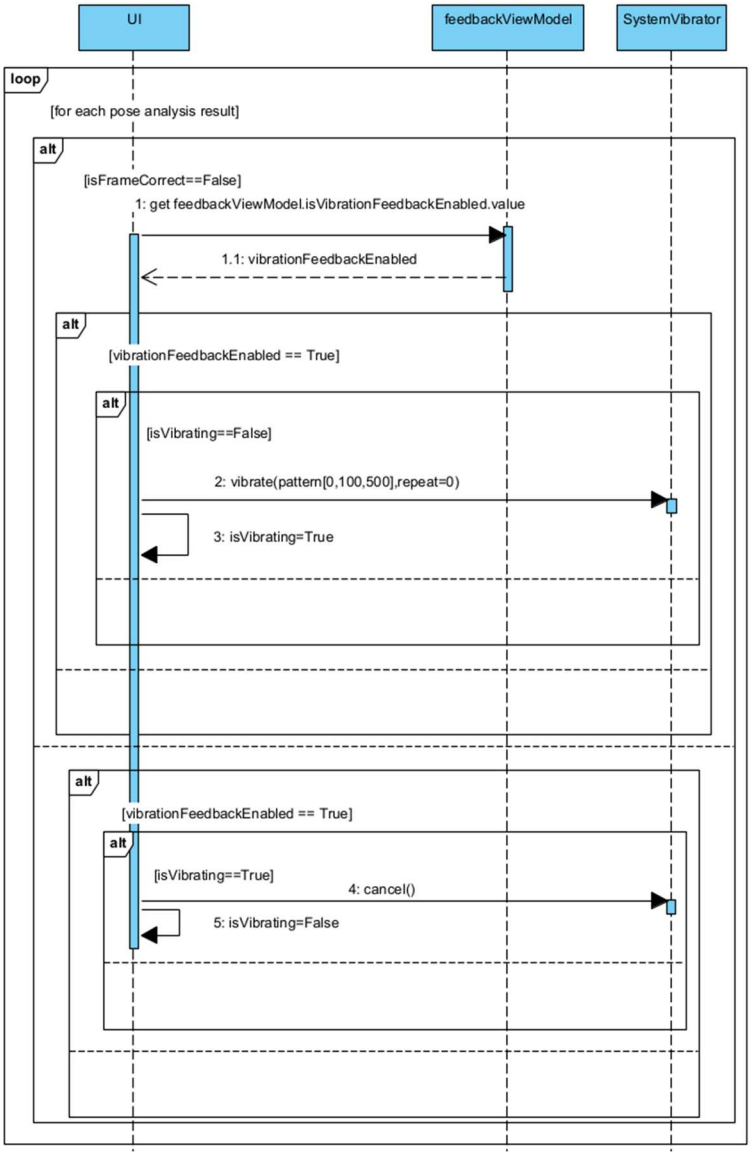


Figure C4. Sequence diagram: vibration feedback system (same implement logic applied to sound feedback).

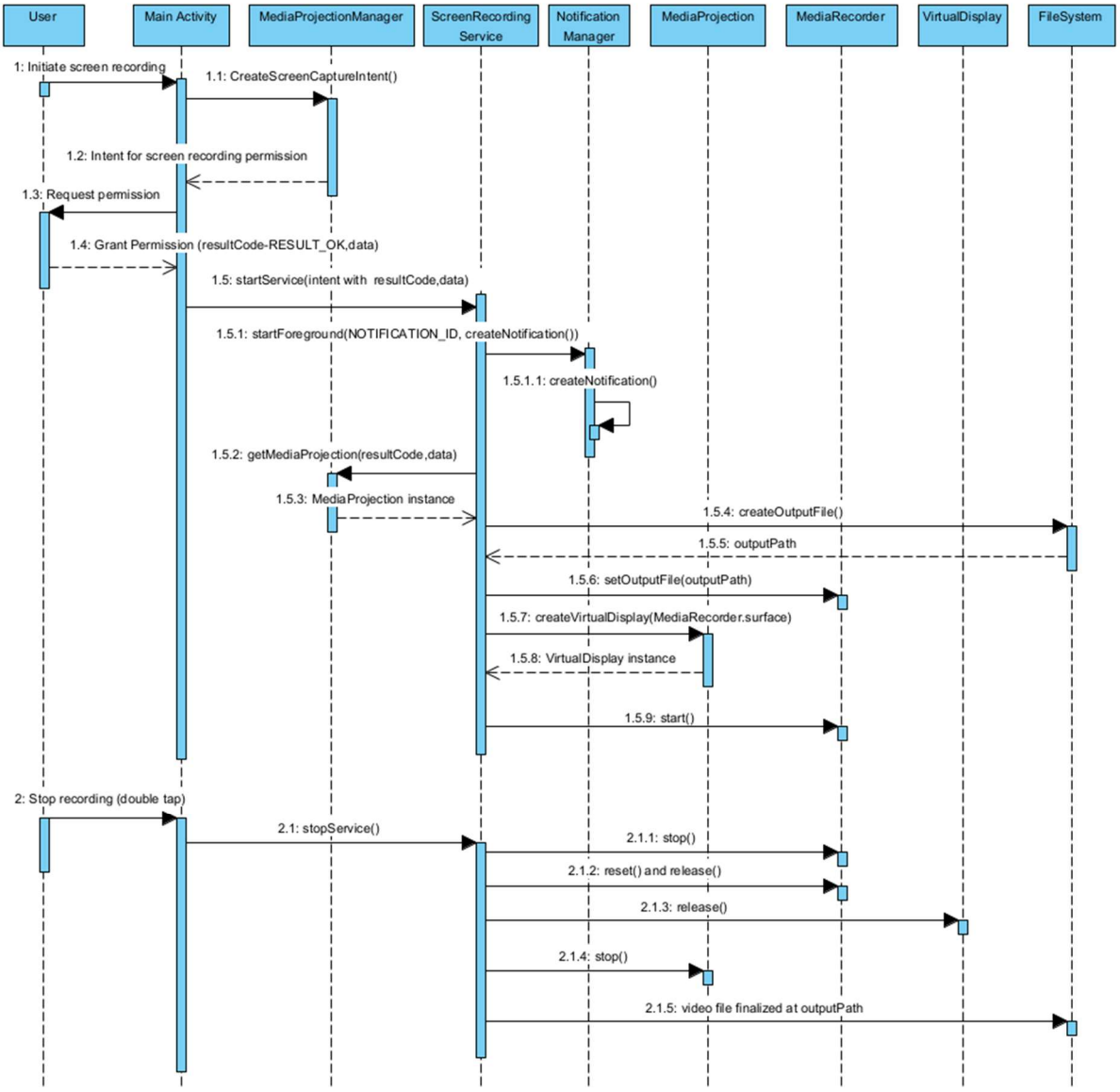
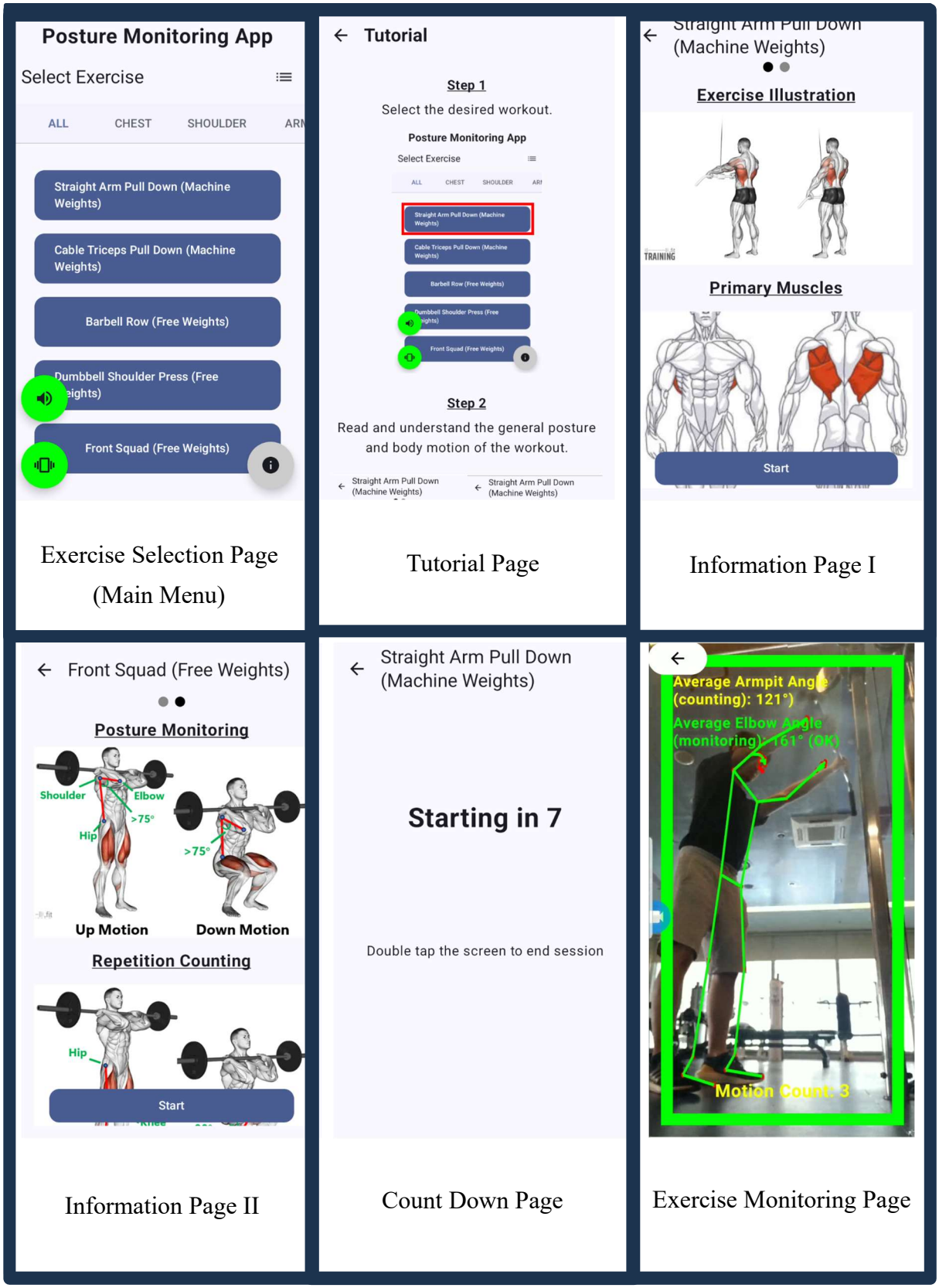


Figure C5. Sequence diagram: screen recording service that is compatible to multiple Android API versions.

Appendix D

User Interface Design



Exercise Selection Page (Main Menu)

Tutorial Page

Information Page I

Information Page II

Count Down Page

Exercise Monitoring Page

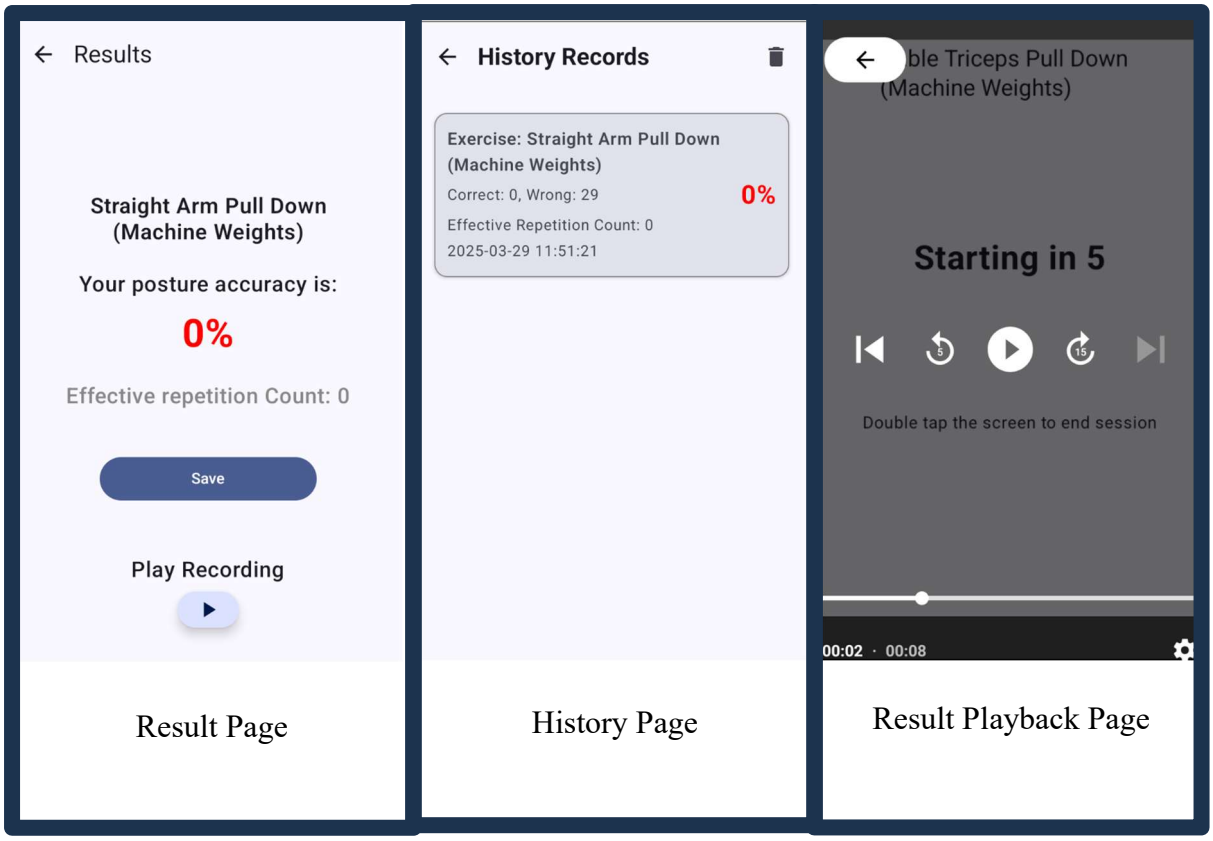


Figure D1. User interface design of each page.

Appendix E

Objectives and specifications of classification tasks

Exercise (Body part)	Posture Monitoring Objectives	Keypoints for Posture Monitoring Task (Condition of correctness)	Keypoints for Repetition Counting Task (Condition of counting)
Straight arm pulldown (back)	Maintaining straight arms	SHOULDER, ELBOW, WRIST ($150° < \theta < 180°$)	ELBOW, SHOULDER, HIP (down: $\theta < 45°$, up: $\theta > 90°$)
Cable triceps pulldown (arms)	Maintaining fixed elbow positions	ELBOW, SHOULDER, HIP ($0° < \theta < 30°$)	SHOULDER, ELBOW, WRIST (down: $\theta > 135°$, up: $\theta < 90°$)
Barbell Row (back)	Maintaining bentness of knees and back	HIP, KNEE, ANKLE ($135° < \theta < 160°$) SHOULDER, HIP, KNEE ($90° < \theta < 160°$)	SHOULDER, ELBOW, WRIST (down: $\theta > 135°$, up: $\theta < 120°$)
Dumbbell shoulder press (shoulders)	Ensuring elbows do not drop	ELBOW, SHOULDER, HIP ($0° < \theta < 165°$)	ELBOW, SHOULDER, HIP (down: $\theta < 105°$, up: $\theta > 165°$)
Incline chest press (chest)	Maintaining bentness of arm	SHOULDER, ELBOW, WRIST ($0° < \theta < 160°$)	SHOULDER, ELBOW, WRIST (down: $\theta < 90°$, up: $\theta > 135°$)
Front squad (legs)	Ensuring elbows are pointed forward	ELBOW, SHOULDER, HIP ($60° < \theta < 180°$)	HIP, KNEE, ANKLE (down: $\theta < 90°$, up: $\theta > 165°$)
Push-up (chest)	Maintaining the flatness of body	SHOULDER, HIP, KNEE ($165° < \theta < 180°$)	ELBOW, SHOULDER, HIP (down: $\theta < 105°$, up: $\theta > 135°$)
Sit-up (abs)	Ensuring knees are overly bent or extended	HIP, KNEE, ANKLE ($60° < \theta < 105°$)	SHOULDER, HIP, KNEE ($60° < \theta < 90°$)
High cable curls	Ensuring arms are	ELBOW, SHOULDER, HIP	WRIST, ELBOW, SHOULDER

(arms)	slightly above shoulder and elbows are not overly extended	(90° < θ < 135°) WRIST, ELBOW, SHOULDER (0° < θ < 165°)	(60° < θ < 135°)

Figure E.1. Posture monitoring objectives and selected angles for classification tasks.

Appendix F

Real-life posture monitoring demonstration

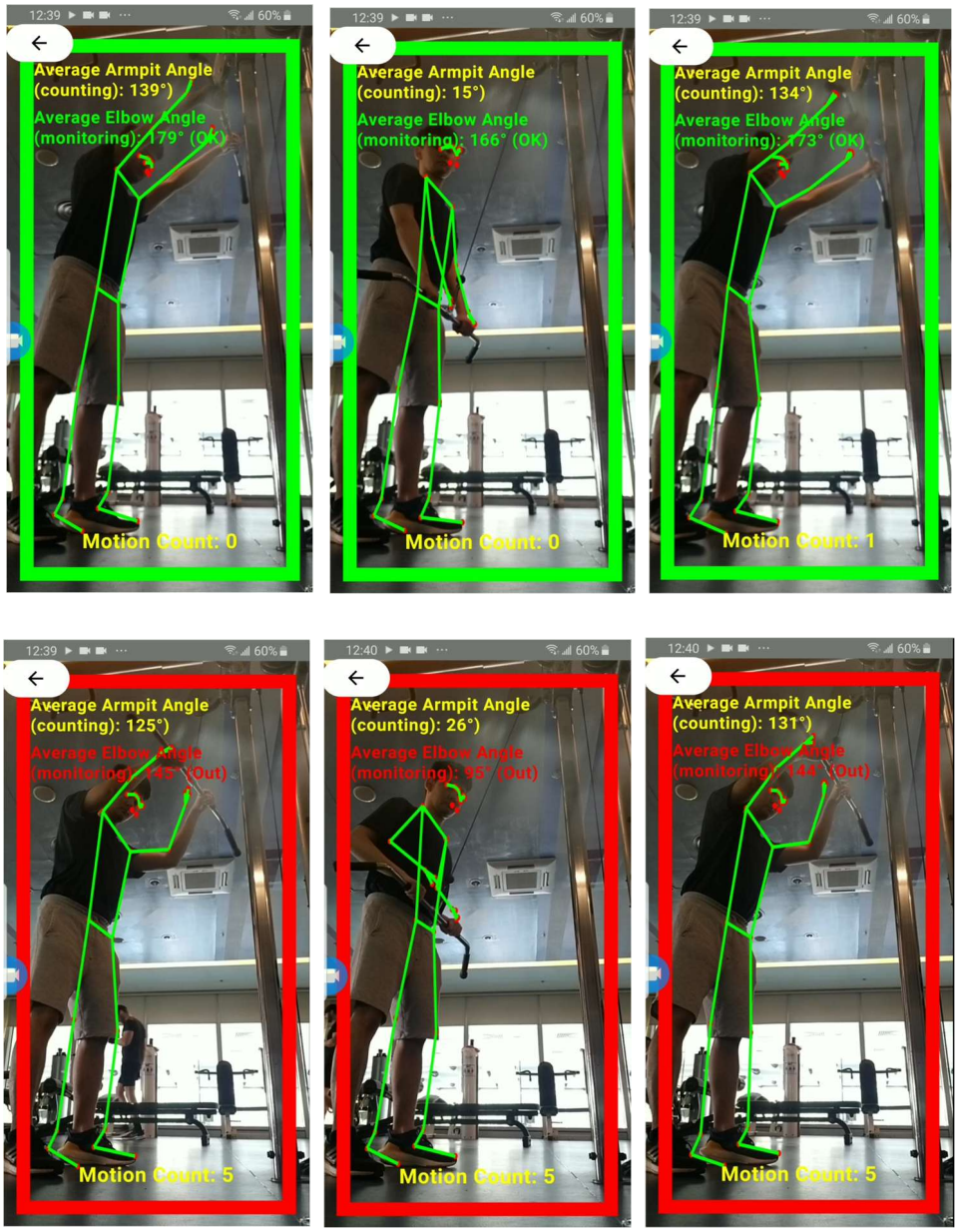


Figure F1. Straight Arm Pulldown Demonstration: monitor results of executing the repetition with correct posture (top) and wrong posture (bottom). Wrong posture include the bending of elbows.

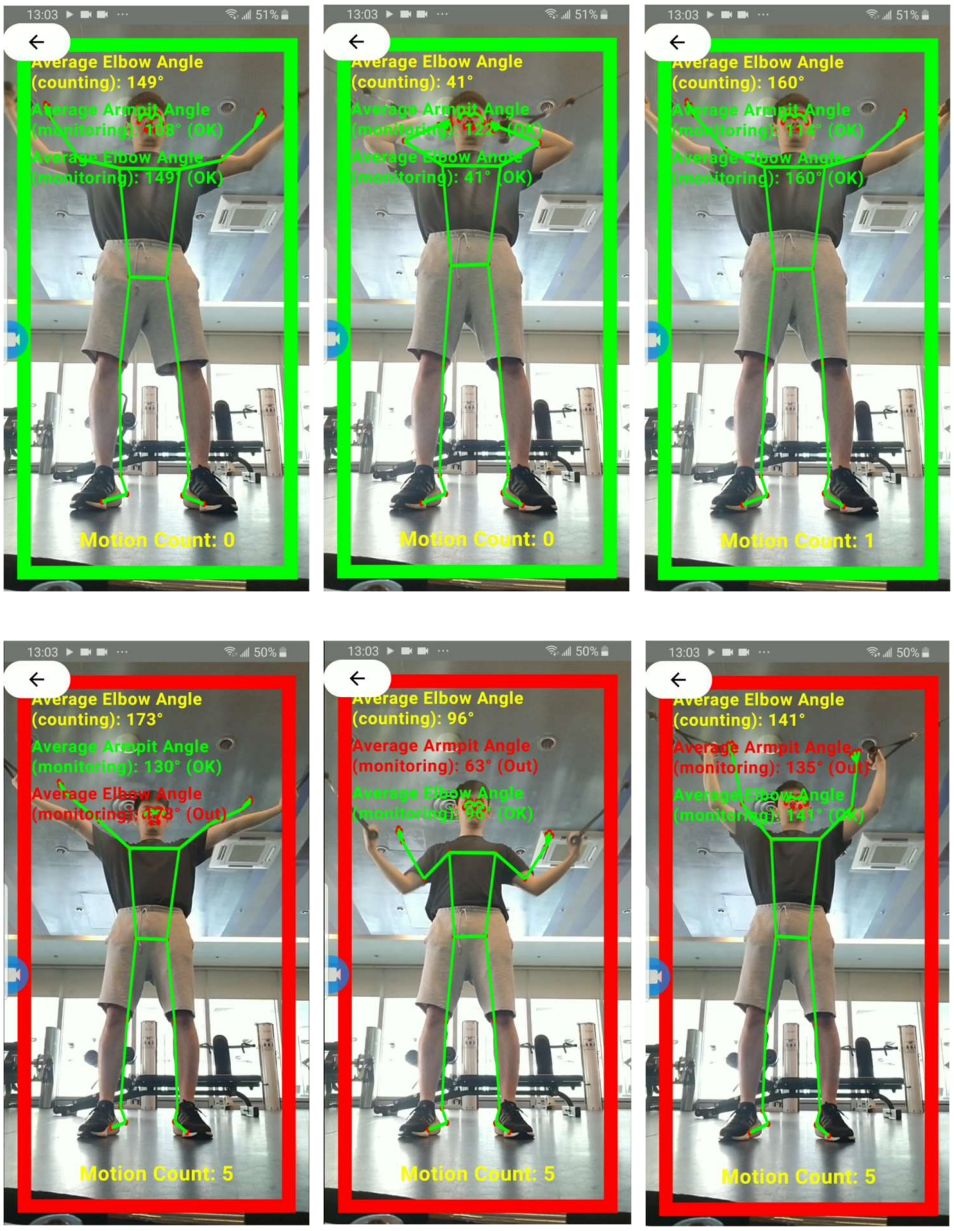


Figure F2. High Cable Curls Demonstration: monitor results of executing the repetition with correct posture (top) and wrong posture (bottom). Wrong postures include full extension of elbows (bottom left), low arm positioning (bottom middle) and high arm positioning (bottom right).

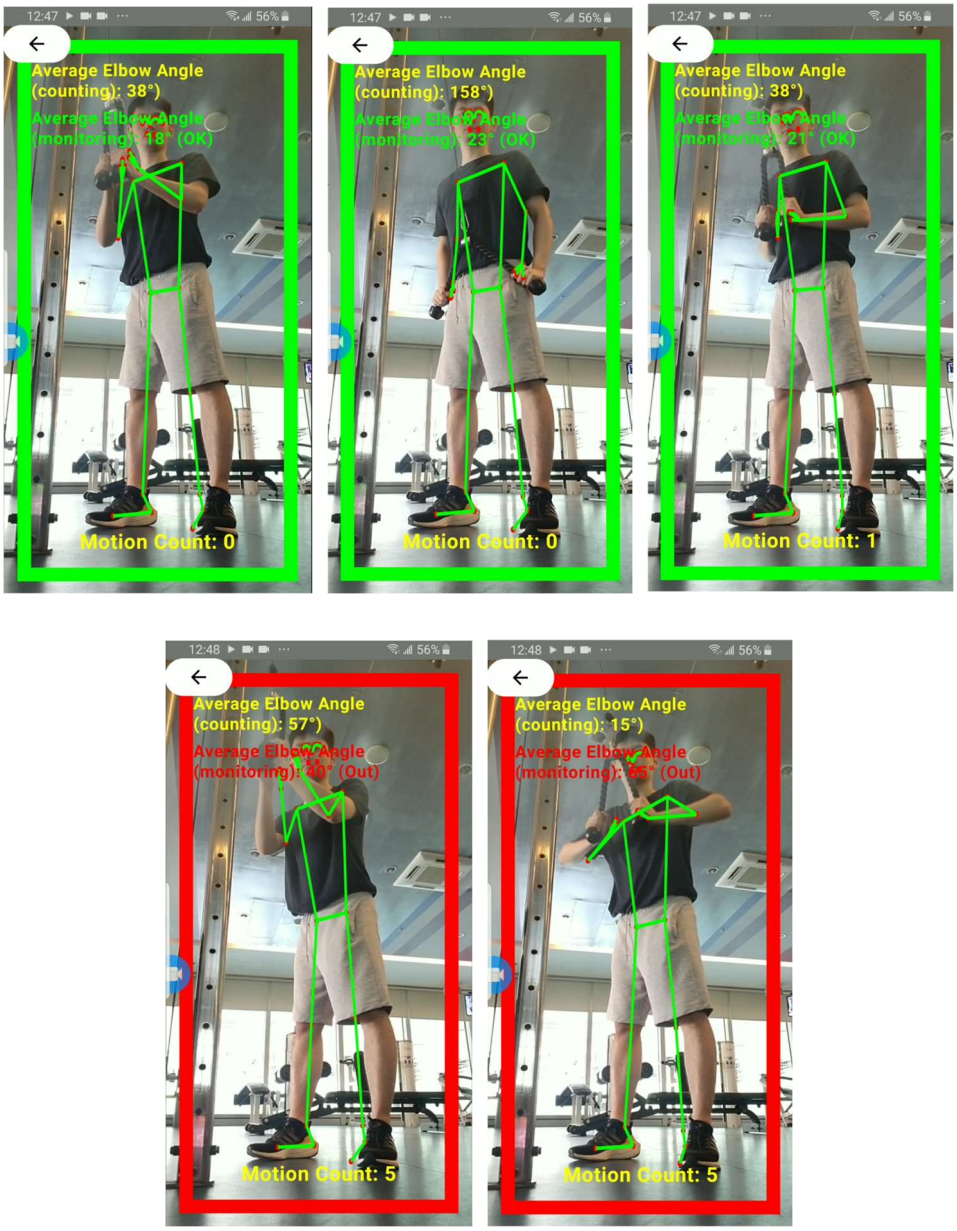


Figure F3. Cable Triceps Pulldown: monitor results of executing the repetition with correct posture (top) and wrong posture (bottom). Wrong postures include the movement of elbows.

Figure F4. Front Squad: monitor results of executing the repetition with correct posture (top) and wrong posture (bottom). Wrong postures include the dropping of elbows.

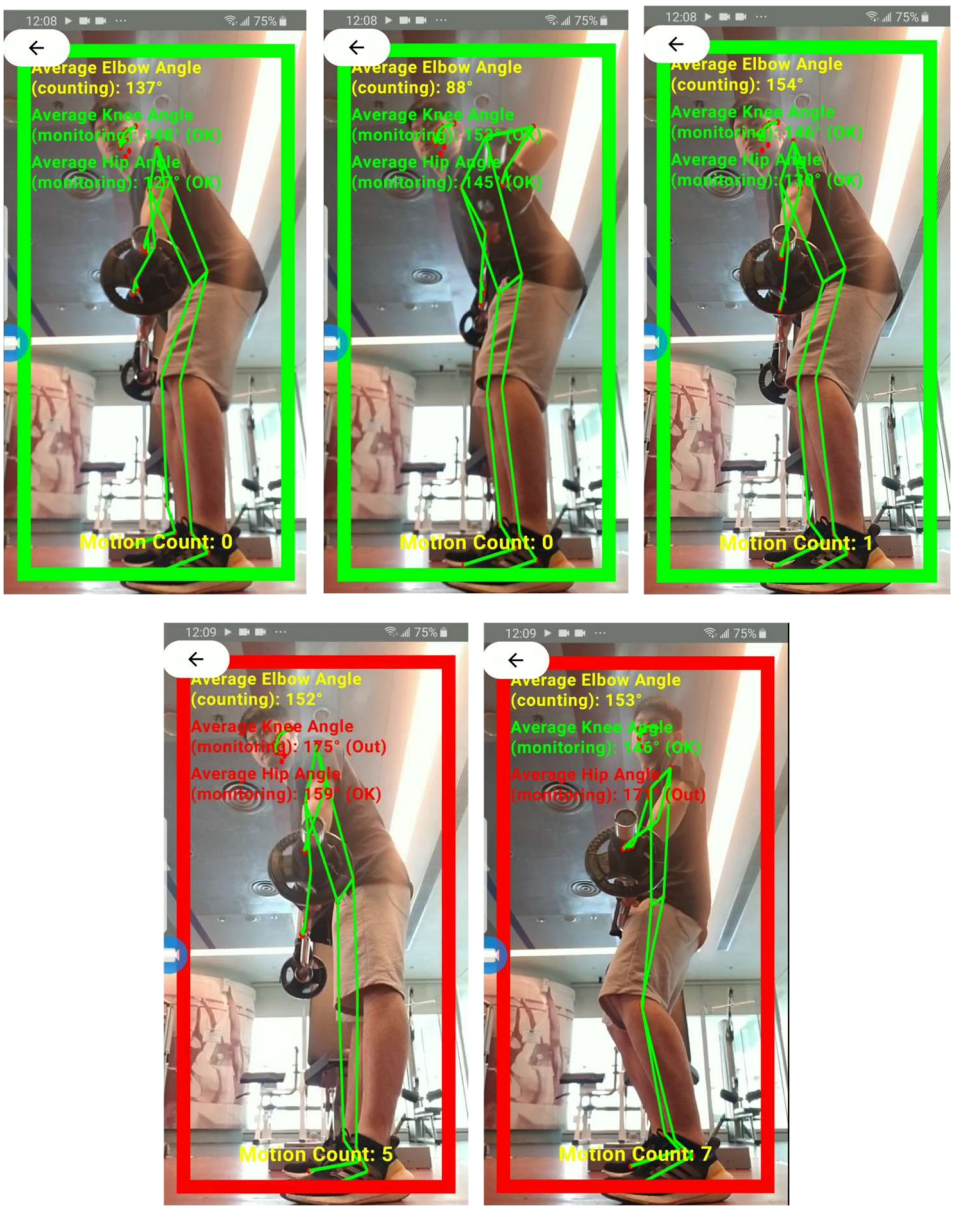


Figure F5. Front Squad: monitor results of executing the repetition with correct posture (top) and wrong posture (bottom). Wrong postures include the full extension of knees (bottom left) and not leaning forward of the body (bottom right).

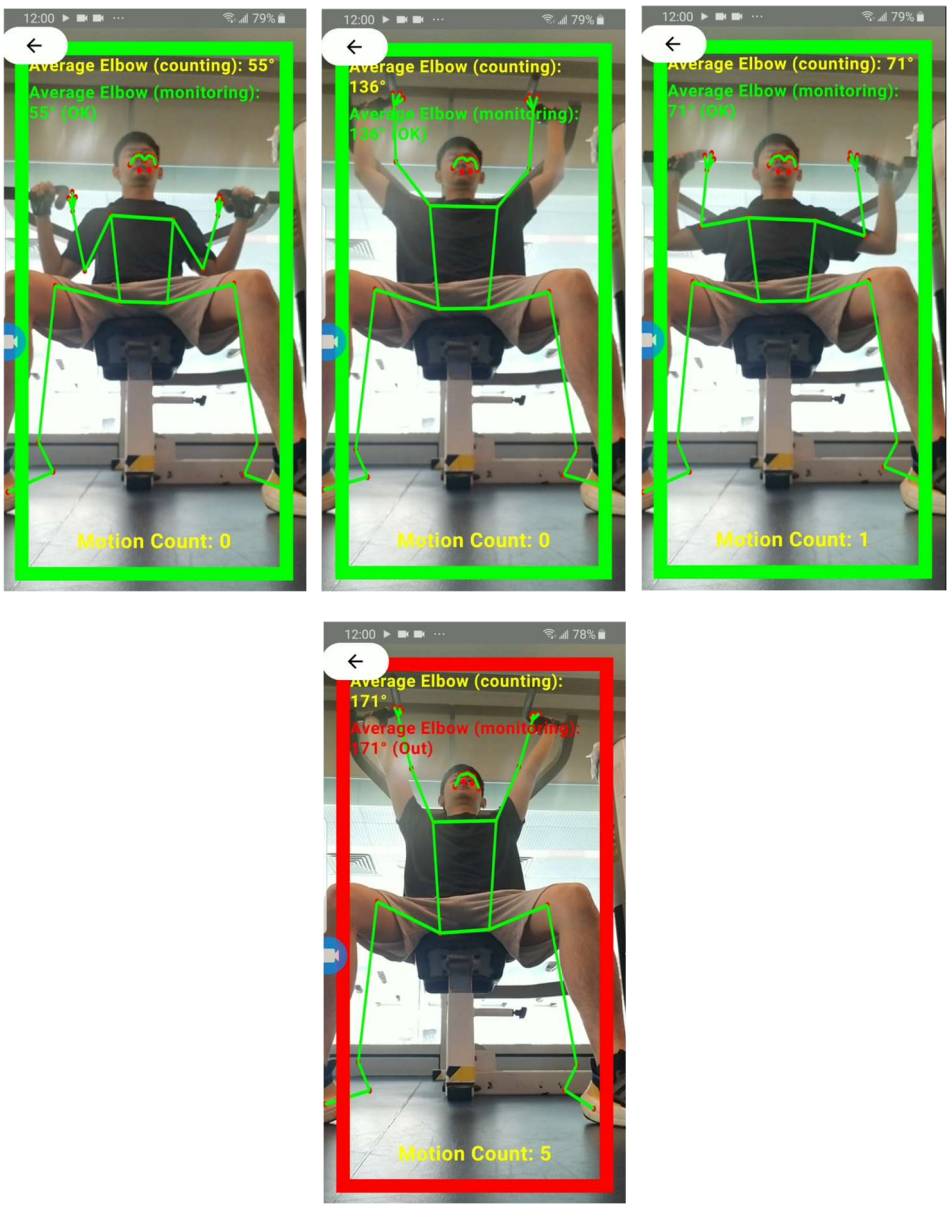


Figure F6. Incline Chest Press: monitor results of executing the repetition with correct posture (top) and wrong posture (bottom). Wrong postures include the full extension elbows.

Figure F7. Dumbbell Shoulder Press: monitor results of executing the repetition with correct posture (top) and wrong posture (bottom). Wrong postures include the dropping of elbows.

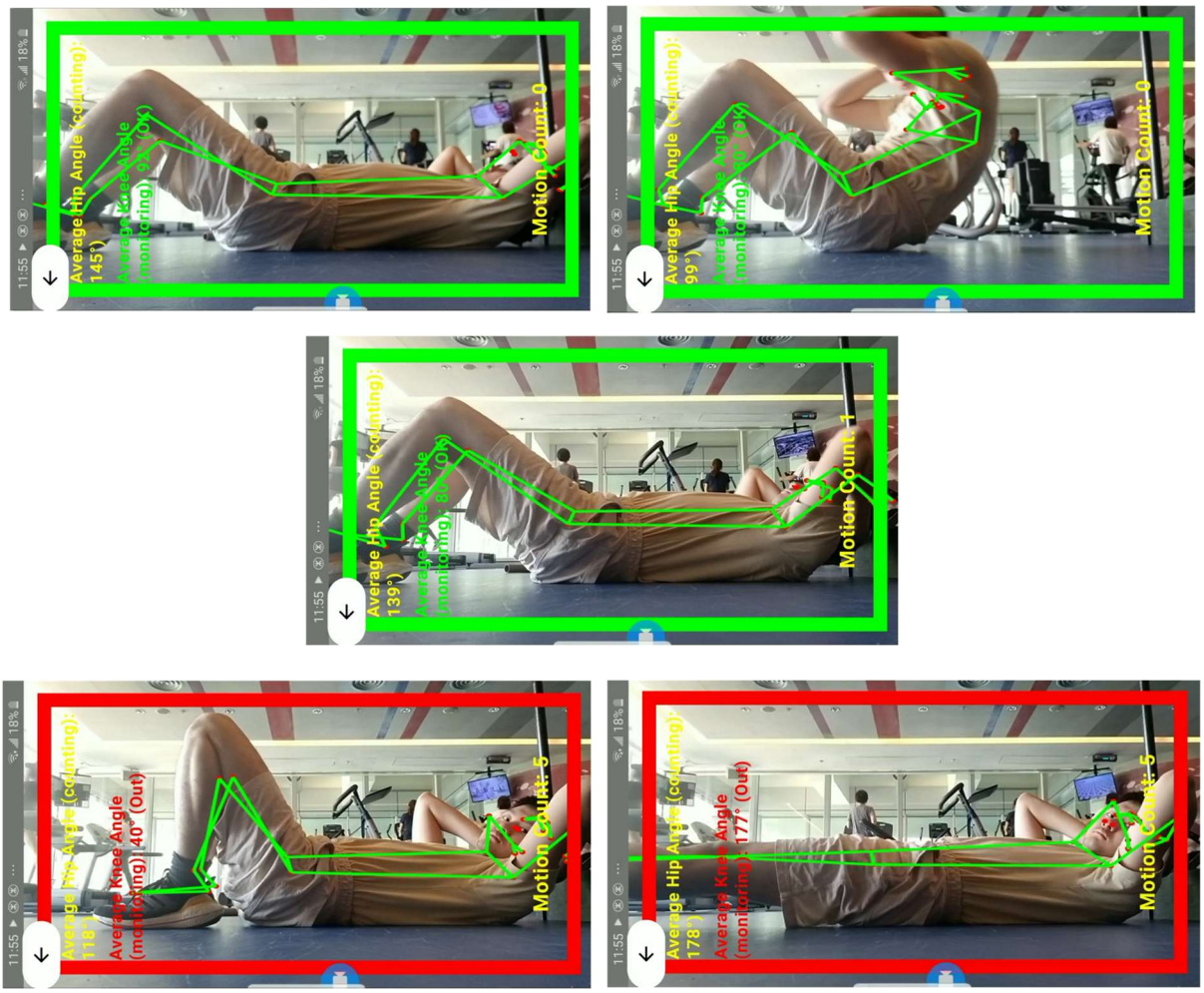


Figure F8. Sit-up: monitor results of executing the repetition with correct posture (top) and wrong posture (bottom). Wrong postures include the over bending knees (bottom left) and over extending knees (bottom right).

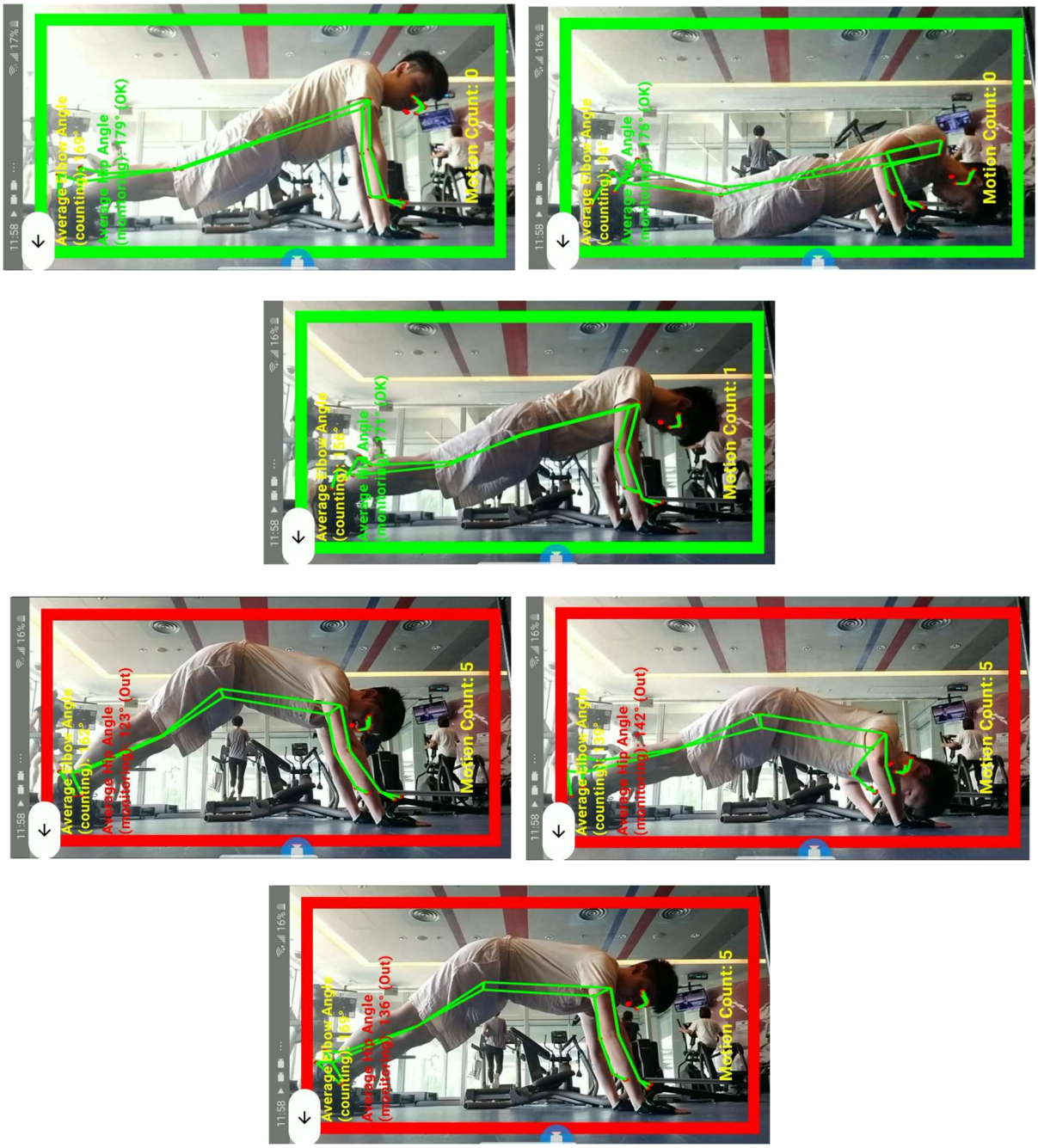


Figure F9. Push-up: monitor results of executing the repetition with correct posture (top) and wrong posture (bottom). Wrong postures include not keeping the body flat.

Appendix G

Functionality testing of application

Testing Period: 2025/02/10 - 2025/02/26				
System / Component / Feature	Expectation	Result (Android API 28, physical device)	Result (Android API 32, emulator)	Result (Android API 35, emulator)
Page Navigation (Navigation)	Able to navigate to *HistoryPage*, *PrePoseEstimationPage* from *MainMenu*.	pass	pass	pass
Page Navigation (Back to Menu)	Able to return back to *MainMenu* from *HistoryPage*, *PrePoseEstimationPage* and *ResultPage*.	pass	pass	pass
Content Generation (Exercise Content Plug-Ins)	Able to dynamically generate exercise-specific buttons based on the existing exercise plug-ins in *MainMenu.*	pass	pass	pass
Content Generation (Educational Content &	Able to load and display exercise-specific educational content in	pass	pass	pass

System Permission: File Reading)	*PrePoseEstimationPage* after click corresponding dynamically generated.			
Real-Time Pose Estimation (System Permission: Camera)	Able to obtain camera permission and display front camera view.	pass	pass	pass
Real-Time Pose Estimation (BlazePose Full)	Able to receive per frame input from front camera and output frame-specific keypoitns	pass	pass	pass
Real-Time Posture Monitoring (Skeletal Overlay)	Able to pinpoint user's joints and display as skeletal overlay on top of camera view	pass	pass	pass
Real-Time Posture Monitoring (Posture Monitoring)	Able to receive keypoints and correctly classify the postures for all the exercises based on defined exercise-specific classification rules in real-time.	pass	pass	pass
Real-Time Posture	Able to receive keypoints correctly	pass	pass	pass

Monitoring (Repetition Counting)	identify up and down motions and perform counting for all exercises in real-time.			
Feedback System (Visual Feedback)	Able to display red border on wrong posture and green border on correct posture in real-time.	pass	pass	pass
Feedback System (Vibration Feedback)	Able to start device vibration when wrong posture is classified, and stop when correct posture is classifier.	pass	pass	pass
Feedback System (Sound Feedback)	Able to start playing alert tone when wrong posture is classified, and stop when correct posture is classifier.	pass	pass	pass
Screen Recording Service (System Permission: Screen Recording)	Able to perform recording after user clicked “allow” on system pop up permission request window.	pass	fail 02/15 remarks: change of Android screen recording permission and security policy	fail 02/15 remarks: same as API 32
Screen Recording Service (System	Able to perform recording after user clicked “allow” on	pass	pass 02/26 remarks:	pass 02/26

Permission: Screen Recording) 02/26 Update	system pop up permission request window.		adapted to new Android security policy	remarks: same as API 32
Summary Report	Able to display summary report after completing posture monitoring session.	pass	pass	pass
History Manager (Saving Records & System Permission: Write File)	Able to save the generated summary report to specified path after user clicked “save” button in ResultPage.	pass	pass	pass
History Manager (Loading Records & System Permission: Read File)	Able to dynamically generate history records and display report information based on the saved summery reports in HistoryPage.	pass	pass	pass
History Manager (Video Playback & System Permission: Media Player)	Able to play the corresponding video recording on ExoPlayer when a history record is clicked in HistoryPage.	pass	pass 02/26 remarks: pass after adapted to the Android security policy	pass 02/26 remarks: same as Android API 32

History Manager (Record Clearing)	Able to clear all previous saved records and screen recordings	pass	Pass	pass

Figure G1. Application functionality testing results on different Android systems. A pass indicates the function has meet the expectation, whereas a fail indicates the function does not meet the expectation or leads to serious errors such as app crash.